A survey of temporal techniques applied toward neural network based continuous speech recognition

Chris D. Love

Defence and Civil Institute of Environmental Medicine

Technical Report

TR 1992-078

July 1992

TR 1992 -78

JULY 1992 DCIEM No. 92-78

A SURVEY OF TEMPORAL TECHNIQUES APPLIED TOWARD NEURAL NETWORK BASED CONTINUOUS SPEECH RECOGNITION

Chris D. Love

Defence and Civil Institute of Environmental Medicine
1133 Sheppard Avenue West
P.O. Box 2000
North York, Ontario M3M 3B9

© HER MAJESTY THE QUEEN IN RIGHT OF CANADA (1992) as represented by the Minister of National Defence

© SA MAJESTE LA REINE DU DROIT DU CANADA (1992) -- Defence Nationale Canada

Neural network (NN) architectures for the recognition of continuous speech are reviewed in this report. Historically, NNs were developed for the recognition of static patterns. To use such networks for speech recognition required that the speech be segmented into chunks such as words or phonemes that could be recognized individually as static patterns. In real speech, the execution of a word or a phoneme depends to some extent on what words or phonemes precede it. These coarticulation effects cause problems unless prior history is used to aid the recognition process. New architectures are being developed to permit the speech stream to be treated as the continuous stream that it is. Segmenting still occurs, which is legitimate, since humans do identify individual words, syllables and phonemes, but the segmentation may be intrinsic to the recognition process. Alternatively, the segmentation may be done by a front-end process that preserves coarticulation effects. Hierarchic structures that recognize events of increasing temporal scale seem to provide the most promising path toward effective recognition of continuous speech.

ACKNOWLEDGEMENTS

I thank the DCIEM and the CSE for providing my with this interesting research topic. In particular, I thank Dr. Martin Taylor for his advice and input given during the development of this work. I also acknowledge the constructive advice of my colleagues at DCIEM. I also thank the librarian of DCIEM's Scientific Information Centre for his help in obtaining materials used in this work.

TABLE OF CONTENTS

Page
COPYRIGHT NOTIFICATION ii
ABSTRACT iii
ACKNOWLEDGEMENTS iv
LIST OF FIGURES vii
LIST OF TABLES ix
LIST OF ABBREVIATIONS AND SYMBOLS x

I INTRODUCTION
Purpose 1
Motivation for Survey 1
Survey Outline 2

II REVIEW OF CONTINUOUS SPEECH RECOGNITION
Introduction 4
Dynamic Programming 8
Level Building and Hierarchical Word Spotting Operation 8
Hidden Markov Models 10
Limitations and Shortcomings 12
Early Neural Network Models 14
Backpropagation 14
Kohonens' Self Organizing Maps 17
Summary 21

III FEATURE PROCESSING
Introduction 22
Temporal Processing 23
The Temporal Issue 23
Biological Processing 24
Temporal Decomposition 28
Temporal Decomposition Model 29
Determination of the Target Functions 31
Determination of the Weighting Factor 33
Determination of the Amplitude Coefficients 34
Speech Segmentation Techniques 37
Linearity and Invariance Conditions 38
Segmentation According to Time 39
Segmentation According to Parameters 40
Summary 41

Page

IV MODIFICATIONS AND ENHANCEMENTS TO THE SPEECH SIGNAL

Introduction 43
Noise Reduction 44
Intonation Adjustment 46
Stress Adjustment 47
Summary 50

V CONTINUOUS SPEECH RECOGNITION USING NEURAL NETWORKS

Introduction 51
Time Delay Neural Network 52
    Temporal Flow Model 53
    Frequency-Time-Shift Invariant TDNN 64
    Block Windowed TDNN 66
    Tempo 2 TDNN 68
    Parsed TDNNs for Varying Phoneme Duration 72
    Enhanced Time Invariance using Pattern Prediction in a TDNN . . . 73
    Time State Neural Network 75
Temporal Backpropagation Neural Network 79
    Focused BP 80
    Adaptive Temporal BP 87
    Forced Simple Recurrent BP 91
Temporal and Hierarchical Neural Networks 99
    The Hierarchical Network 97
    Adaptive Dynamic and Associative Memory Model 108
Summary 113

VI SURVEY DISCUSSION

Introduction 114
Discussion 114

VII CONCLUSIONS AND RECOMMENDATIONS

Conclusions 123
Recommendations 126

REFERENCES 127

LIST OF FIGURES

Figure **Page**

2.1 Overview of how contemporary recognizers attacked the problem of continuous speech recognition 7
2.2 Connected word recognition using level building 9
2.3 State diagram of a Markov process 11
2.4 Three layer backpropagation architecture 15
2.5 Representation of an artificial neuron and its logistic activation function . . 16
2.6 A Kohonen self organizing map architecture 18
2.7 Updating region of the Kohonen output layer 19

3.1 Biological neuron and synapse descriptions 26
3.2 An excitatory action potential 27
3.3 An idealized sketch of a target function 32
3.4 Target functions resulting from various models for three segments of speech 33
3.5 Plots of the target functions obtained by temporal decomposition of three curves 36
3.6 Iterative modification of the analysis window size and position 36
3.7 Plots of the speech waveform for 'Joe brought a young girl' and the various target functions determined by temporal decomposition 37
3.8 Description of the linearity and invariance condition 39

4.1 Noise cancelling approach 45

5.1 Architectural overview of the TFM *place* experiment 57
5.2 Delay link profile for *place* experiment of the TFM 58
5.3 Response of [b, d, g] output units (units 58 - 60) of place network optimized on 30 training repetitions to an MSE of 0.3% to samples of training data . . 59
5.4 Architectural overview of the TFM *manner* experiment 61
5.5 Delay link profile for *manner* experiment of the TFM 61
5.6 Response of output units (units 44 - 48) of manner network optimized on ten training repetitions to an MSE of 0.35% to samples of training data 63
5.7 TDNN architecture for six phoneme class problem 65
5.8 Frequency-time-shift-invariant TDNN (FTDNN) architecture for the six phoneme class problem 64
5.9 Block-windowed TDNN architecture for the six phoneme class problem . . 67

Figure **Page**

5.10 Two Gaussian shaped input windows with different weights, delays, and widths as used by the Tempo 2 network 69
5.11 A graphical explanation of the learning rules for delays and window widths 70
5.12 Splitting a neuron connection due to oscillations in the Tempo 2 network 71
5.13 Modified TDNN architecture used for effectively capturing varying phoneme durations 73
5.14 Proposed recurrent neural network architecture for enhanced pattern prediction 74
5.15 TSNN architecture used to perform the six-phoneme classification problem 76
5.16 An improved TSNN architecture by using modularity in the hidden layer 78
5.17 Abstract characterization of the temporal recognition task 81
5.18 Unfolded BP architecture 83
5.19 Activity trace and context implementation used in focused BP 85
5.20 Mean performance on the reversed regular verb task as a function of learning epoch for the focused and fully recurrent architectures 86
5.21 Spatialized temporal recognition 88
5.22 Architecture of a modified temporal BP network 89
5.23 The SRN and FSRN 92
5.24 The architecture and results of the first experiment between the SRN and FSRN 96
5.25 The architecture and results of the second experiment between the SRN and FSRN 98
5.26 An enhanced temporal module 100
5.27 A chain of temporal modules for an extended sequence 102
5.28 A hierarchical network for extended temporal recognition 103
5.29 Output of LSNs of M1 through M3 resulting in a correct ERN response 104
5.30 Output of LSNs of M1 through M3 resulting in an incorrect ERN response 105
5.31 A hierarchical temporal network indicating module outputs of level-1 and level-2 for a correct temporal sequence consisting of patterns $\{I_1, I_2, I_3\}$ 106
5.32 A hierarchical temporal network indicating module outputs of level-1 and level-2 for a partial temporal sequence consisting of patterns $\{I_{2p}, I_3\}$ 107
5.33 A set a four snapshots in time (T1 - T4) of the activation of an utterance 110
5.34 Parallel and modular architecture of the proposed Man-Machine architecture system 112

LIST OF TABLES

Table **Page**

5.1 Phoneme Recognition Results For Various TDNN Architectures 75
5.2 Phoneme Identification Performance Using Time State Neural Networks . . 79

LIST OF ABBREVIATIONS AND SYMBOLS

AI	Artificial intelligence
ANN	Artificial neural network
ARPA	Advanced research projects agency
ASR	Automatic speech recognizer
BP	Backpropagation neural network
CPN	Counterpropagation neural network
CSR	Continuous speech recognition/recognizer
DARPA	Defense advanced research projects agency
DP	Dynamic programming
DTW	Dynamic time warping
Hz	Hertz (cycles per second)
HMM	Hidden Markov model
LMS	Least mean square
LPC	Linear predictive coding
M/s	metres per second
Ms	milli-second ($1e^{-3}$ seconds)
MV	milli-volts ($1e^{-3}$ volts)
MSE	Mean squared error
NN	Neural network
RMS	Root mean square
S	second(s)
SVD	Singular valued decomposition
TBPNN	Temporal backpropagation neural network
TD	Temporal decomposition
TDLAR	Temporal decomposition log area ratio parameters
TDNN	Time-delay neural network
TSNN	Time-state neural network

I
INTRODUCTION

1.1 Purpose

?

The objective of this literature survey is to investigate the temporal issues surrounding using neural networks based continuous speech recognition (CSR) systems. In particular, this survey investigates neural networks and techniques that deal with temporal variability and connectivity in natural speech as opposed to those techniques that work well with temporally segmented and corrected speech. These two approaches are contrasted to decide the probable benefits and difficulties in performing CSR ~~with respect to~~ in approach, ease of implementation, and effectiveness.

1.2 Motivation for Survey

This study is motivated by the need to increase the understanding of neural network based CSR systems in the area of temporal recognition. A better understanding of temporal recognition of continuous speech using artificial neural network architectures results from this investigation.

Many recognizers have attempted to segment speech discretely, but speech does not occur in this fashion. Speech is temporal, which means that speech cues overlap in time and support the presence of larger linguistic units including adjacent phonemes, words or sentences. "Difficult-to-determine aspects of speech include its temporal nature and lack of divisions between words in continuous speech" [Dayh90]. Recent neural network models that use temporal integration can do CSR because, with other temporal processes, they can find the correct word boundaries, which reduces the pattern recognition problem. This is a good solution to the problem of performing continuous speech recognition because past architectures attempted to segment speech discretely but this resulted in poorly determined utterance boundaries. In turn, this led to bad pattern representations, which ended in unsuccessful recognition. Thus, we are also motivated to reexamine the segmentation problem and then apply the results effectively to NN architectures. In doing so, the NN architectures must be modified to take full advantage of *temporal speech knowledge*, which encompasses many facets of speech including physiological (coarticulation), phonetic, acoustic, context (at higher constructs), time and frequency-time shift invariance, scaling, and some prosodic systems.

1.3 Survey Outline

This survey has four sections. The first section (Chapter II) discusses past CSR techniques, including descriptions and limitations of Hidden Markov Models (HMM), Dynamic Programming (DP) and early artificial neural network (ANN) architectures. These ANNs had many similar limitations to the noncognitive systems. The discussion focuses on how these models address temporal variability and segmentation and whether they use correction/ enhancement techniques to overcome the severe limitations resulting from the absence of temporal speech knowledge.

The second section (Chapters III and IV) gives the problem's foundation. Chapter III describes the temporal issues surrounding CSR, an overview of biological temporal processing and concludes with an indepth discussion on temporal segmentation and decomposition techniques. Chapter IV discusses ways to enhance the temporally segmented speech samples so that better recognition results. This discussion includes noise reduction, and stress and intonation adjustments.

The third section (Chapter V) is the focus of the survey. This section describes NN systems that use temporal processing to perform CSR. These descriptions include only neural network based architectures, their methodologies and architectures, and their experiments and results.

The last section (Chapter VI and Chapter VII) relates the survey results. Chapter VI contrasts the view of treating speech in a strict temporal and continuous manner, i.e., provide continuous speech input to a NN, to one that temporally decomposes, enhances, and corrects the speech segments prior to recognition. The main difference is that in the first case all aspects of segmentation and recognition is the network's responsibility, while in the second case, the network is only responsible for recognition; the utterance has already been segmented. Also, the main ideas described in the survey are brought together here to show any underlying common characteristics of temporal recognition neural networks. Chapter VII presents the survey conclusions. These comments include the success in reaching the survey objectives and a summary of the important temporal network approach results. Recommendations concerning the use of temporal speech recognition NN systems follow.

II
REVIEW OF CONTINUOUS SPEECH RECOGNITION

2.1 Introduction

The primary objectives of this chapter are to introduce and define the continuous speech recognition problems and review four popular contemporary models. This last point provides a reference point upon which judgement of the new contemporary systems will be established.

This chapter begins by looking at natural speech recognition problems. Following this, two popular contemporary models, dynamic programming, and hidden Markov models are described because both are good at performing continuous speech recognition, given their inherent limitations. Finally, two NN models are examined that do not explicitly perform temporal recognition but are successful anyway at performing CSR, namely backpropagation and Kohonens' self organizing maps.

Continuous temporal speech recognition means recognition of naturally spoken utterances. Some neural networks have special architectures and algorithms that help to solve the problem of natural speech recognition. The algorithms are special because they simulate a temporal environment, which implies that the network can produce a solution based on a time dependent input series. Before these approaches are described some common problems and their solutions are discussed, which early researchers encountered when developing natural speech recognition systems.

What is Wrong with the Past Techniques?

"To increase competence, connectionist learning strategies should *build* on existing distributed knowledge rather than trying to *undo, ignore* or *relearn* such knowledge" [Waib89a]. This statement sets the tone for the following discussion that concerns the problems of some existing systems.

In [Chur87] Ken Church describes his views on connected speech recognition problems, which are summarized below. First, many past systems never attempted continuous speech recognition, instead they stayed with isolated recognition systems. The isolated recognizers though, got accuracies in the high nineties (>99%). Essentially his reasons for this tendency toward isolated recognition systems were that word boundaries are difficult to find and the acoustic parameters of continuous speech exhibit much greater variability, depending on the context, than isolated speech. Therefore, "any attempt to extend the philosophy of isolated word recognition systems and recognize the utterance in a continuous stream becomes an exercise in futility" [Chur87]. He concludes the discussion by saying that, "The technique needed [to recognize continuous speech] becomes one of analysis and description rather than classification (moving away from pattern recognition paradigms toward hierarchical systems, i.e., systems in which component subparts are recognized and grouped together to form larger and larger linguistic units . . .)" [Chur87].

"Most CSR systems are very dependent on locating word boundaries, which are among the most poorly and inconsistently articulated portions of speech" [Bris86]. Alex Waibel says that continuous speech presents added difficulties because word boundaries are either unknown or are very difficult to find [Waib88]. Pronunciation in connected speech is sloppier than in isolated words and coarticulation effects are more severe. Coarticulation often *smears out* otherwise crisp information. Short words are especially affected: *the* can be reduced to [ə] and *and* to [n]. "Hence, just when segmentation forces us to lean more heavily on phonetic information, that same information becomes less reliable" [Pars86]. Most recognizers are not selective in what they analyze since they give *equal* attention to *all* acoustic details. It can be argued that many recognizers are simply pattern matchers that could recognize bird whistles with the same amount of difficulty as speech. The hidden levels of linguistic analysis and regard for prosodic features are missing. These can provide cues to syntactic structures (words, sentences, etc.). "Human beings recognition of speech consists of many tasks, ranging from the detection of phonemes to the high-level understanding of messages. We do not actually hear all speech elements; we realize this easily when we try to decipher foreign language or uncommon utterances. Instead, we *continuously* relate fragmentary sensory stimuli to contexts familiar from various experiences and unconsciously test and reiterate our perception at different levels of abstraction" [Koho88]. "Finite state grammars used in recognition systems have been known for decades to fail to characterize the phaseal groupings of words in continuous sentences thus leading to inefficient models. To AI proponents, it is strange that devices disregard the fact that we *do* have a working example system that very successfully performs at the speech recognition task, and that system is the *human perceiver*" [Bris86].

Investigation of more biologically plausible approaches to CSR that incorporate techniques such as temporal integration, auditory phonetics, and linguistic categorization processes should therefore be pursued. Even the rule governed variabilities in sound structures that natural English contains should be considered. "Most recognizers do not even consider phonetic, linguistic, auditory, temporal, or perceptual methods. No single system currently contains even a major fraction of what is known about speech production, acoustics, hearing and perception within human listeners and linguistic categorizations" [Bris86]. These ideas are shown graphically in Fig. 2.1. Many current recognizers that use signal processing and pattern matching techniques are very sensitive to insignificant and irrelevant variabilities including speaker noise, rate of articulation, acoustic environment (and changes in it), microphone movement, etc..

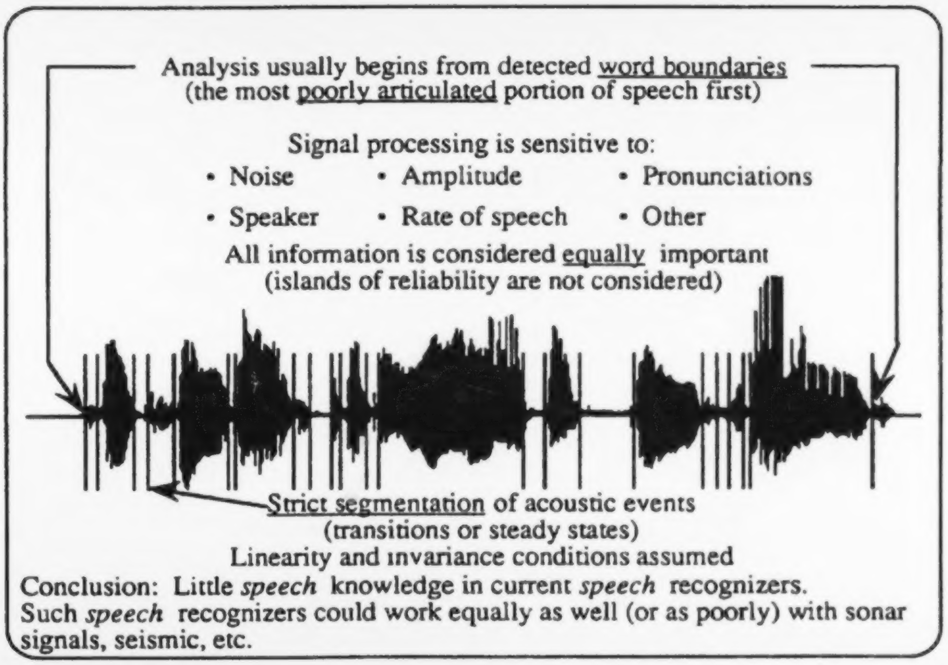
Analysis usually begins from detected word boundaries
(the most poorly articulated portion of speech first)
Signal processing is sensitive to:
• Noise
• Amplitude
• Pronunciations
• Speaker
• Rate of speech
• Other
All information is considered equally important
(islands of reliability are not considered)
Strict segmentation of acoustic events
(transitions or steady states)
Linearity and invariance conditions assumed
Conclusion: Little speech knowledge in current speech recognizers.
Such speech recognizers could work equally as well (or as poorly) with sonar signals, seismic, etc.

Fig. 2.1 Overview of how contemporary recognizers attacked the problem of continuous speech recognition (after [Brns86]).

Many, if not all, of the apparent problems can be overcome by using more built-in knowledge about the language. Understandably, the human brain is far superior to any existing artificial speech recognizer so we are not hoping to do recognition at this level of performance, but it is still necessary that CSR system developments begin to use or increase the use of more speech knowledge based techniques so that progress continues.

In summary, to resolve the worse suppositions and errors found in past CSR systems additional speech knowledge must be used. It is no longer acceptable to attribute stress, coarticulation, and noise as reasons for why systems fail to recognize difficult speech because many new systems are solving these problems [DBMC88, BaMA89].

2.2 Dynamic Programming

A more successful connected word recognition technique is word spotting. Word spotting involves continuous speech recognition at the word level and it uses dynamic programming (DP) techniques to perform the pattern matching. An instance of this approach, called level-building, follows.

2.2.1 Level Building and Hierarchical Word Spotting Operation

One implementation of word spotting is *level building*. This is a "level-by-level time-warping algorithm which drastically reduces the number of possible paths to be evaluated . . ." [Pars86]. A level usually defines the area in the search graph that matches a particular word template to a speech segment representing that word. Setting the template boundaries at the word level prior to the DP search helps word level recognition. Because recognizing word boundaries is a very difficult and inaccurate process a built-in error margin is present that reflects this inaccuracy. Searching the entire template library yields likely candidates for each unknown word. Thus as the number of system vocabulary templates grows so too does the search time. If real-time recognition is not an issue, then this approach may be viable for particular CSR applications. The search keeps multiple word candidates at each level. At successive levels, testing former level paths with the current level paths help decide the identity of the current speech segment. A cost function determines the less likely paths, which are then pruned. After the search reaches the highest level, only one connected path remains. The DP algorithm then backtracks to find this optimum path and while doing so, also determines the various speech components along the way using the template library. A graphical illustration of this process is given in Fig. 2.2.

An immediate disadvantage of word spotting is that the recognition takes place at the *word* level. To recognize the English language would require an enormous database. For large vocabularies this approach does not appear feasible unless the vocabulary is tractable.

Limiting the data set, for instance to the digit set, provides some advantages because digits are typically spoken in bursts so the signal length is short.

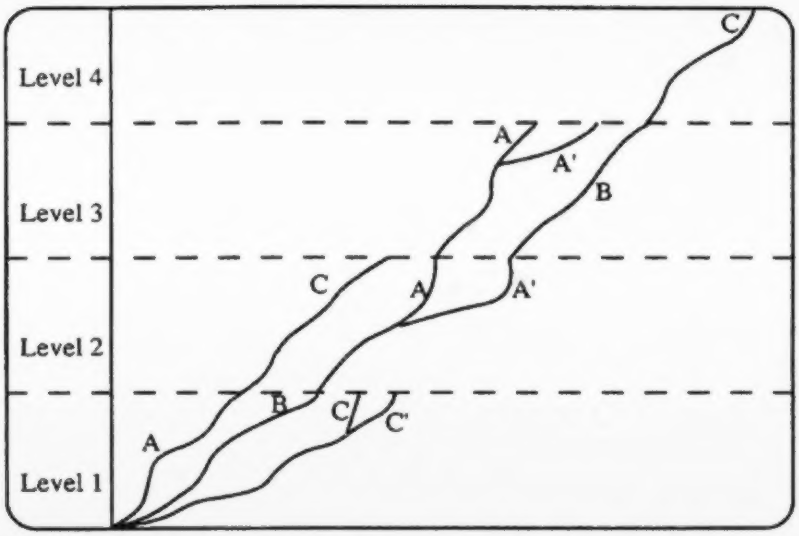

Fig. 2.2 Connected word recognition using level building. Each level corresponds to the identification of possible words (using dynamic time warping). Connected streams identify possible connected sequences. Levels do not necessarily need to be equal length (after [Pars86]).

Finally, having the network learn the transitions between two digits may enhance recognition since coarticulation would then be represented in the network, and this is not unreasonable to implement because the number of combinations is not too excessive. As stated earlier, there is a vocabulary expansion problem. If many words are to be recognized memory storage may become a limiting factor.

Templates typically consist of isolated words that may have been recorded in isolation. For example, *going* may have been generated by recording it in isolation or it may have been electronically *chopped* from a sentence. Due to the way humans speak and their effects on words within a sentence the two variations of *going* will produce different templates and thus, give varying results [FeKi91]. This is because humans tend to speak isolated words much more carefully than those produced in a steady stream. If the programmer decides that templates chopped from naturally spoken sentences would be more useful, the problems of word boundary location must be considered.

The last disadvantage concerns temporal processing. Although this approach uses soft decisions it does not take advantage of *any* temporal processing. Language is a temporal process. Phonetic cues support the evidence of more complex acoustic events by way of temporal integration. Briefly, temporal integration sums acoustic activations over time to decide if a particular event has occurred. The time that it observes the inputs is determined by its time constant.

2.3 Hidden Markov Models

Hidden Markov modelling (HMM) is very popular for both isolated and continuous recognition. This recognition technique makes use of a stochastic model of speech production and compares favorably with DP approaches while requiring only a fraction of the computational load. HMM was probably the most general framework used in speech recognition until the resurgence of NNs. Hidden Markov models provide automated training, efficient temporal alignment, and temporal matching.

The HMM is a state machine and as such, is in only one of several states at any given time. Each state generates many random outputs. Transitions between these states occur in a discrete fashion, have random probabilities, and permit the model to deal with subtle ambiguities of word pronunciations. The sequence of activity occurring as the model moves between various states, with each state generating an output, constitutes the recognition of a particular utterance. An example of an utterance HMM is given in Fig. 2.3. Circles represent states and the lines and arrows show direction, connectivity and transitional probability. Intuitively the model's *mechanism* represents the vocal tract (although it is not strictly identified with it), the *states* represent the particular positions of the vocal apparatus, and the *outputs* of each state represent the various observable acoustic parameters. Each word in the desired vocabulary can be represented by a model similar to that given in Fig. 2.3.

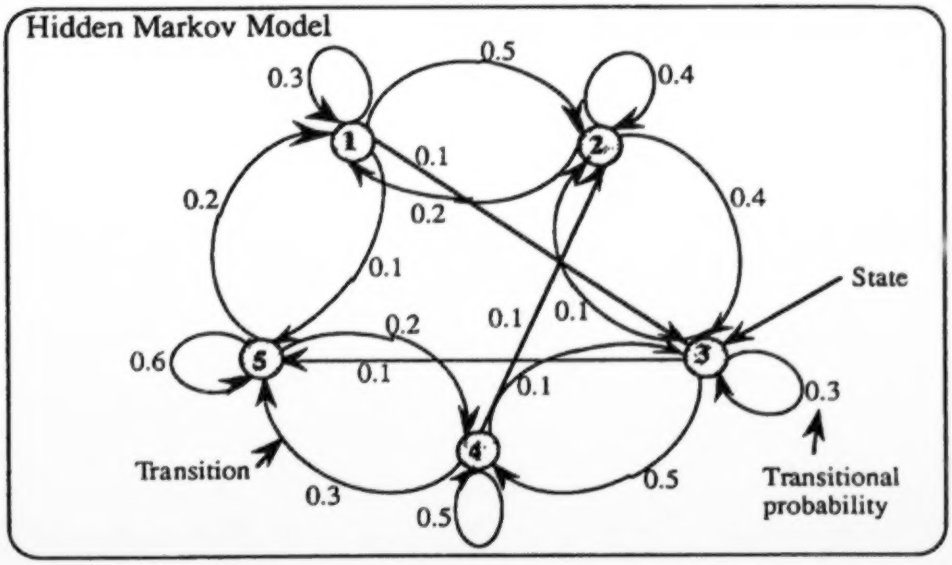

Fig. 2.3 State diagram of a Markov process. The lines show the connections and their allowable random interstate transitional probabilities. The circles represent states in the Markov process (after [Pars86]).

The recognizer decides the identity based on the discrete outputs resulting from the state-to-state transitions. Its task is to find the word model that most likely generated the observed sequence of outputs. An HMM is hidden because the recognizer can only see its outputs, not the word models, or more importantly, the flow in the word model as it progresses from state-to-state. To implement a model like that given in Fig. 2.3, assume the model to start in some random state, q_i and then proceed to the next state, q_j. The transitional probability between two states, a_{ij}, represents the probability of going from

$$\mathbf{A} = \begin{bmatrix} 0.3 & 0.5 & 0.1 & 0.1 & 0.0 \\ 0.2 & 0.4 & 0.4 & 0.0 & 0.0 \\ 0.0 & 0.1 & 0.3 & 0.1 & 0.5 \\ 0.2 & 0.0 & 0.0 & 0.6 & 0.2 \\ 0.0 & 0.1 & 0.1 & 0.3 & 0.5 \end{bmatrix} \quad a_{ij} = P\,\{\text{transition from } q_i \text{ to } q_j\}$$

state q_i to q_j. Assuming there are N states then leads to the compact format of the N x N matrix, $\boldsymbol{A}$. Since the outgoing transition probabilities sum to one, so too does each row of $\boldsymbol{A}$. The system can manage state omissions or repetitions because it is nondeterministic, which is useful due to known variabilities in pronunciation. If there is a finite set of $\boldsymbol{M}$ possible outputs $\{z_i\}$ then with each state q_i we associate a vector of $\boldsymbol{M}$, $\boldsymbol{b}$, where $b_j = \{z_j\}$. Similar to $\boldsymbol{A}$, an N x M matrix, $\boldsymbol{B}$, represents all the possible state outputs, where the ith row is the vector $\boldsymbol{b}_i$. Since output probabilities sum to one, so too does each row of matrix $\boldsymbol{B}$. Training the recognizer involves constructing a word library. Every word model has an associated state number, transition matrix, $\boldsymbol{A}$, initial state probabilities, $\boldsymbol{p}$, and output matrix, $\boldsymbol{B}$. In operation, the system starts off in a random state. Given a speech parameter, the model proceeds to the next state using maximum likelihood theory. At each state an output is generated, stored, and later processed using maximum likelihood theory to decide the word model.

2.3.1 Limitations and Shortcomings

This model assumes a finite number of outputs, and in CSR, a method for reasonable z_i selection must be determined beforehand. Rabiner et al. chose to solve this problem by applying vector quantization. They chose this technique because it automatically performs the clustering that is conveniently associated with the z_is. HMM, like DP require an equal number of models to words. In Dynamic Time Warping (DTW) the models are templates while in HMM they are discrete Markov models. In either case it is necessary to have as many of these as the number of words in the vocabulary; for a large vocabulary the system gets very big, very fast.

In HMM, some training parameters pose serious problems due to *underflow*. This involves a scaling problem of certain parameters in the training algorithm; during training certain α and β's become increasingly small. To avoid this, a complex dynamic normalization process is used. Also, the components of both HMM matrices must be estimated. This entails many parameters for a seemingly small problem. For instance, given an alphabet with 64 distinct characters means that B will have 320 elements with 335 parameters that may be used to fit each letter's pronunciation in the alphabet. "The amount of recognition data needed to form reliable estimates of recognition parameters grows roughly ***exponentially*** with the number of parameters, which implies that long training sessions will be necessary to obtain reliable models" [Pars86].

The key HMM weakness is that there is typically not enough data to train the model to yield reliable results. One author said ***woefully inadequate*** with respect to the amount of training data. Parsons [Pars86], on the other hand, felt that the major problem of HMM concerned vector quantization, which is necessary in obtaining the set of z_is.

DARPA suggested four key limitations of the HMM:

"(i) Internal model structure is not learned but must be pre-specified before training. Model builders must specify sub-word models, provide phonemic transcriptions of words, and develop rules to describe allowable types of phonological variation caused by word boundary effects and changes in talking style and dialect.

(ii) Coarticulation is modelled poorly. Coarticulation caused by nasals or laterals may extend over many acoustic segments. This is not easily captured with a single-order Markov assumption.

(iii) Poor acoustic-phonetic modelling results in confusions between similar sounding words.

(iv) Supervised labelled training data is required to train high-level allophone and word models. The abundance of unsupervised speech data that is readily available, thus, can't be used" [DARP88].

In keeping with the survey objectives (as partially mentioned in DARPA (ii)) is the presence of temporal processing in HMM, of which, very little exists. Nor is there any inherent speech knowledge being used in these models. In continuous speech recognition HMMs cannot compete with a temporal based neural network CSR system for the above reasons since NNs can better deal with these problems.

2.4 Early Neural Networks Models

Early neural networks mean those networks that do not incorporate temporal processing. NNs provide the necessary higher computational rates for CSR using parallel architectures and processing approaches. The investigation and discussion into these networks reveal that although NNs are great generalizers they are not much better than, for instance, HMMs unless changes are made to the NN architectures and algorithms. These enhancements include the incorporation of temporal processing into the network. If NNs can adapt their internal parameters over time to optimize performance, self-organize their architecture to optimally represent data and dynamically capture new data structure representations as they arrive, they will outperform HMMs. For now though some models (and their limitations) are examined that do not have these additional changes. In particular, standard backpropagation (BP) and Kohonens' self organizing maps are described. Both are used extensively in the area of speech recognition.

2.4.1 Backpropagation

BP [McRu89] is a supervised feed forward neural network that uses gradient descent to adjust its weight space. Many applications use BP for a variety of tasks, including speech recognition. What limitations does BP have with respect to continuous temporal speech? Also, does BP incorporate, either in its algorithm or architecture, any higher level speech knowledge? BP in its *vanilla* format, as shown in Fig. 2.4, does not lend its architecture to performing time dependent integration and thus cannot recognize continuous speech well. It can recognize continuous speech if the pattern is spatialized over the input layer, time normalized, and centered over the input layer in the same way as it was trained. This process, yet, is difficult to conduct and introduces much error. The result is poor recognition.

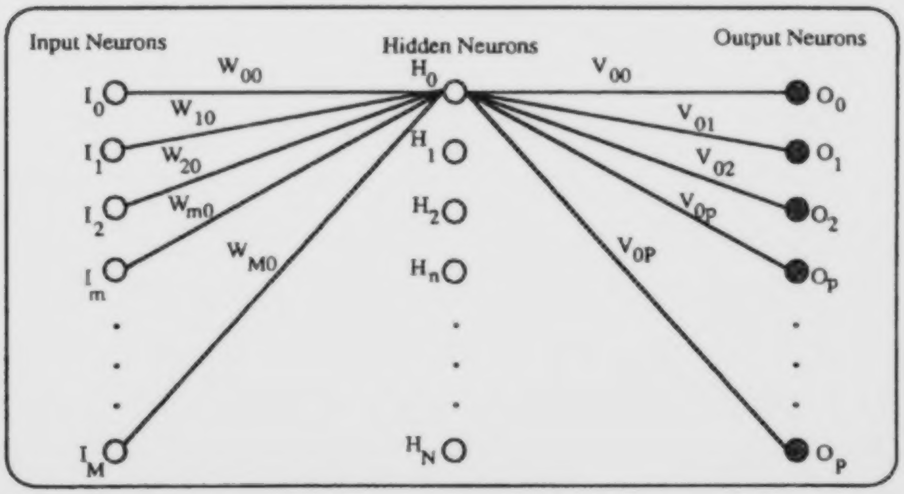

Fig. 2.4 Three layer backpropagation architecture. The hidden and output layers do not show the biases. The architecture is fully interconnected between layers. No lateral connections are used nor does any feedback exist, therefore, no temporal integration exists. The result of this is that temporal recognition of sequences is not possible.

The operation of an artificial neuron is shown in Fig. 2.5. Reconfiguring the BP architecture, which is discussed later, gives the NN the ability to perform temporal recognition. BP commonly uses a sigmoid function to generate the neuron's output. It uses the negative of the error gradient to modify the weight space during training. Configuration and optimization of the network and its parameters, including momentum and learning rate, is difficult. Biases, which are used in the hidden and output layer, shift the sigmoid activation function favorably along the summation axis, improving the discrimination boundaries.

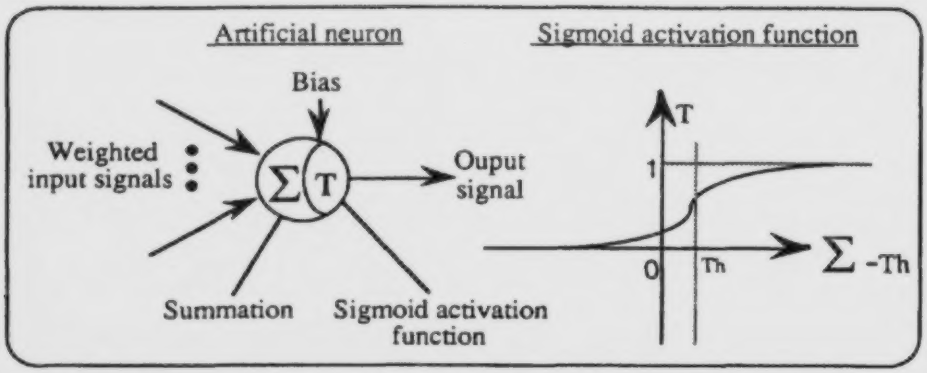

Fig. 2.5 Representation of an artificial neuron and its logistic activation function.

Standard BP does have a limitation though; it cannot do temporal integration. Nor is BP biologically plausible because in natural neural systems a signal cannot travel backward in an axon as in BP's *backpropagation* step. Also, there are many different kinds of natural specialized neurons that have complex dynamics whereas the BP neurons are very simple. Next, natural neurons use many chemicals that excite and inhibit neural receptors. These chemicals allow the neurons to *learn* and *forget*. Forgetting is an important idea in learning. Artifacts, which are undesired attraction basins formed in the space due to noise in the data set, will erode over time because the noise anomalies that are used to form them occur only occasionally. The desired patterns will remain in the weights since these are refreshed often enough that the memory fading is won out by the learning process. Too many hidden neurons cause the network to memorize the relevant features of the data and have poor performance on new data. Similarly, having too few neurons will prevent the network from retaining all the training data features and thus, also result in poor recognition on new data. Next, time-shift invariance to the data must be implemented. Presenting the data earlier or later in time or, compressing or expanding it, which is common in speech, results in lower recognition performance. This network ability is necessary in the continuous recognition domain. Another consideration, which is important in speech recognition systems (of which BP is incapable), is the ability to learn new patterns after training. In other words, the retention of additional patterns is not possible after training unless these patterns are added to the existing data and the network is retrained. This makes vocabulary expansion difficult since training times for nontrivial problems are usually on the order of days, even using a 68040 processor and math coprocessor (68882).

In summary, standard BP cannot perform temporal recognition. It does not incorporate enough speech knowledge to make it viable for CSR. It provides excellent results on isolated patterns given that the time dependent variable (scaling and normalization) has been controlled, either through preprocessing or careful data acquisition. There are many architectural and parameter dependent problems that also make training difficult. As will be shown later though, with certain modifications BP can successfully deal with temporal patterns.

2.4.2 Kohonens' Self Organizing Maps

It appears in the biological case that by arranging neurons and their connectivity strengths in a particular fashion, a better representation of the input results. That is, spatial order is necessary for an effective representation of model information. Kohonens' feature maps do this using only a one or two dimensional framework, which automatically implies a data reduction because of drastically reduced dimensionality. This is the motivation behind self organizing maps - to organize the network automatically to represent the data set regularities.

Kohonen [Koho88] presented an algorithm that produced what he called *self organizing* feature maps, which are similar to those found in the human brain. Figure 2.6 shows that the network input layer receives a sequential series of continuously valued input vectors. During this process, there is no output vector - this is unsupervised learning.

After sufficiently many trials "weights will specify cluster or vector centers that sample the input space such that the point density function of the vector centers tends to approximate the probability density function of the input" [Lipp87]. Another way of thinking about this is to consider that interconnected adaptive units have the ability of changing their responses so the location of the cell in a network, where the response is elicited, becomes specific to a certain feature of the data set.

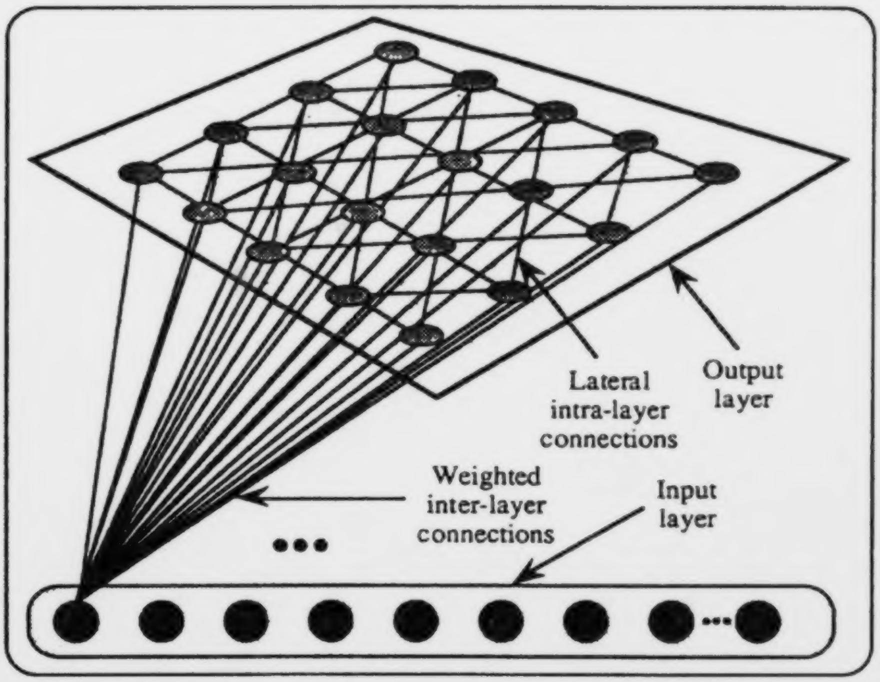

Fig. 2.6 A Kohonen self organizing map architecture (after [Lipp87]).

Thus, the weights become organized *mathematically* such that topologically close neurons react similarly to an active node, i.e., neighbors become involved with an active neuron, but to a lesser degree, possibly in a Gaussian fashion. The sum of the square weights emanating from an output node back to its inputs are normalized. The output node with the smallest Euclidean distance is found by generating the dot product of the input vectors with their respective weight vectors, again looking from an output node back to its connected input nodes. Finding an output node with minimum Euclidean distance then turns into a problem of finding an output node with a maximum value because the components are closer in direction (i.e., larger magnitude components) with the weight vector resulting in a larger value. After identifying this node (called the *winner*), it and its surrounding neighbors are updated by increasing the weight strengths of the nodes in the update region, which moves the space vectors toward the input. The neighbors are those nodes that lie in a symmetrical and topologically local region to the *winner*. Initially, the update region is large but it shrinks uniformly as learning progresses. The updating strengths are usually not identical across the region, but drop off according to a Gaussian or Mexican hat function. An example of this updating process is given in Fig. 2.7.

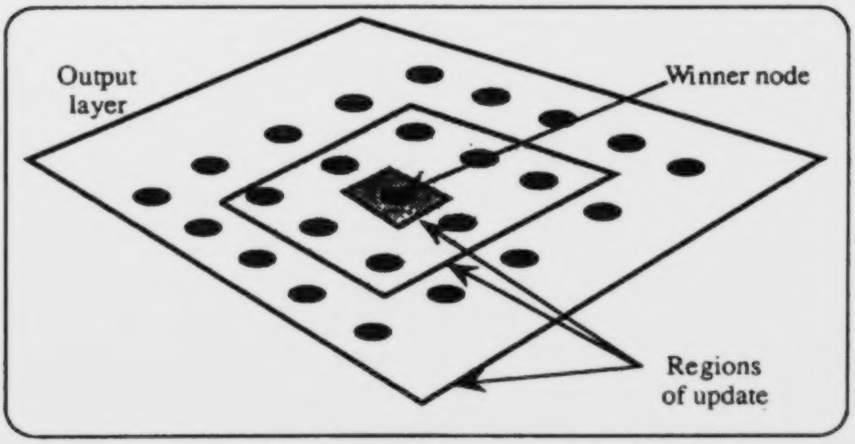

Fig. 2.7 Updating region of Kohonen output layer.

Besides the nodes connected between input and output, there are also lateral nodes interconnected amongst the output nodes (see Fig. 2.6). There are many 2-D layers of neurons with lateral connections that densely interconnect the human brain. These lateral connections come from outputs of neighboring units within the output layer. The lateral connections help the network's performance in various ways. First they help by biasing nodes when a particular pattern type is presented. This is done by neighboring nodes partially exciting the correct target node, thus lowering the activation threshold. Second, it helps cluster like categories. Similar patterns will be mathematically close based on a distance metric. The use of a mexican hat function in the lateral connectivity process inhibits dissimilar patterns and therefore makes the activation threshold higher and less likely to become active. Last, the region of attention about the correct node shrinks in distance as learning progresses, thus smaller regions experience the prize of increased weights. This connectivity and updating strategy improve recognition. The particular locale of the lateral coupling depends on the ratio of excitatory to inhibitory connections, to the central excitation function, which is the lateral coupling strength. Thus a wider region of participation provides increased positive feedback while a narrower region of attention leads to increased negative feedback. The best results come from beginning with a large updating region and later shrinking it according to some linear function. The learning rate (gain factor) used in updating the weight shrinks according to a nonlinear function and, when the learning rate falls below a threshold, the learning process is terminated.

In summary, Kohonen's self organizing map is useful in the following ways: first, it performs automatic clustering of data that, when considered in the speech domain, turns out to be useful since utterances are also clustered according to vocal position, formants, etc.. Next, the data reduction done by this NN is useful for speech recognition because better discriminatory boundaries may result if the right amount of uniqueness (typically from three to five dimensions - formants) is retained. Remember that this network is taking a high dimensional signal (speech) and mapping it into a low dimensional space. Using the existing structure of the network, it does not appear that any form of temporal recognition is possible, unless consecutive, overlapping segments are spatialized over the input space. One other alternative is to modify the network in a modular and hierarchical manner that would provide feedback throughout the different levels of hierarchy. More of this approach is described in Chapter V.

2.6 Summary

DP, HMM, BP, and Kohonens' self organizing maps were described and evaluated. Limitations of these models were given. Many of these models are only good at isolated, limited vocabulary, speech recognition. Yet, when the problem domain was extended to temporal continuous speech recognition, the models' architectures and algorithms fell short. The problem of temporal recognition is addressed in the next chapter and models that deal with this added dimension in the recognition domain are described.

III
FEATURE PROCESSING

3.1 Introduction

The objectives of this chapter are to introduce and describe temporal processing and temporal decomposition, which are essential to CSR. Biological temporal processing is briefly mentioned as motivation toward the use of temporal decomposition. This chapter focuses on temporal decomposition, which segments speech considering coarticulation effects. A discussion on improved classical segmentation techniques is given, which provides some interesting ways to compete with, or help the temporal decomposition approach.

To recognize speech it must first be characterized by a set of rules. These rules help when designing new speech recognition systems. Features are those observables that characterize speech and provide clues about the utterance's identity. Some models use amplitude, pitch, formants, and spectral coefficients from the speech signal as features. These models also may dynamically compress or expand, time align, filter, and segment speech into identifiable *chunks* before recognition. Feature processing involves knowing what to look for in the speech, ways of extracting the properties, and finally representing them in a way that is useful for recognition. Some new (brute-force) neural networks are fed continuous frame spectral input and do both segmentation and recognition using the same network. The primary interest is not so much in these networks but ones where the network has the benefit of a front-end feature based processor that decomposes continuous speech. This was extremely difficult to do until now because of coarticulation and the limited effectiveness of known segmentation techniques.

3.2 Temporal Processing

In temporal processing the problem of time confronts us. This section describes what temporal processing is and the problem as it relates to continuous speech recognition. Also, a brief biological description of temporal processing is given to motivate the use of temporal processing in the neural networks and to show how it helps in continuous speech recognition.

3.2.1 The Temporal Issue

Phonemes uttered in isolation are the most meaningful basic sounds one can make. Serious problems arise when computers try to recognize naturally composed syntactical structures including words or sentences. These problems arise partly because of the structure of continuous speech. When we speak, we do not speak one phoneme crisply, and then begin to speak the next. Instead, we make meaningful sounds through modulation of our vocal articulators to cause some effect, i.e., to talk [Huck90]. In doing so, we slur phonemes together. The degree of slurring and other factors contribute to the recognition problem. This *overlap*, or coarticulation, is caused by the vocal articulators moving from their present arrangement to a new one [NiFa89]; only about 10 phonemes/s are produced by an adult [RaSc78]. Nor do the articulators stop to reset themselves prior to beginning the next phoneme. During speech, the vocal articulators move from one target to the next, where each target characterizes a particular phoneme. When the articulators arrive sufficiently close, they change their direction and head for the next target. The targets are not easily identifiable since they never reach their destination; the trajectories, or signatures they follow though, are easier to identify. It is important to realize that *speech is not disjoint.* One sounds flows *into* the next, not *after* the next. "The variability caused by coarticulation is systematic; i.e., particular sounds modify adjacent sounds in a *regular* manner" [NiFa89]. This suggests that when we build a speech recognition system we should try to work *with* the presence of coarticulation and use some form of temporal than discrete segmentation to decompose speech. The other important realization is that much information is encoded in the *sequence* of phonemes, i.e., a sequence of syntactically arranged phonemes (or words) carries more (contextual) information than randomly ordered ones. At the phoneme level certain phoneme combinations can be excluded since they are not phonetic in English. Thus if the recognizer senses such a combination it will know an error has been made and possibly try to correct this error by evaluating higher constructs such as tri-phone or word boundaries. If this type of scheme is successfully implemented, high-level recognition will result. To implement an approach of this nature, especially if it is based on natural speech, an efficient technique is needed to separate the components (phonemes or words) correctly since coarticulation is present. This is where the necessity for a temporal processing technique arises.

In summary, to deal with continuous speech recognition, the issues surrounding the production of continuous speech must be addressed and attempts must be made to take advantage of the language structure and characteristics instead of spending effort in tailoring the speech to fit the system's abilities. Most important to this survey are the issues of temporal variability and connectivity in natural speech. This cannot be separated entirely from the other issues such as coarticulation or segmentation. To begin the investigation into temporal processing a brief description of a biological neurons characteristics and operation is given to help understand natural temporal processes.

3.2.2 Biological Processing

This section describes structural and functional properties of biological neurons that help to'explain temporal processing and to indicate the assimilation amount of ANNs to biological systems. Because ANNs are derived from their biological equivalents, it seems appropriate to attempt to incorporate as much physiological speech knowledge as possible into the network. This view is motivated by evolution because, throughout the many years of our development we have genetically evolved improved learning strategies, memory processing methods, and linguistic processes. "In the biological case, learning takes place within an initial structure that arises through genetic expression" [Watr90]. Many of our basic skills are passed on through genes. Unfortunately, the vast majority of this structure is still unknown to us and the current state of neurophysiology is incomplete, therefore we must rely on techniques that may not always be completely biologically plausible to optimize an ANN's structure.

Structural and Functional Properties of Biological Neurons

The brain is the most sophisticated and powerful computing device known. Although it calculates certain things far slower than a desktop PC, it far excels the capabilities of even the most powerful computers for other tasks including vision and language recognition. The neuron is the basic information processing unit of the brain. There are approximately 1 000 billion neurons in the human brain. Each neuron may be connected to as many as 10 000 other neurons. A neuron has a cell body called a *soma*. Connected onto the soma are many *dendrites* that provide the soma with electrical impulse signals. Leading from the soma is an *axon* that later branches often. It carries an impulse signal (~ 100 mV lasting 1 ms at speeds of up to 120 M/s), otherwise known as an action potential, without attenuation down the axon. These action potentials can occur with a frequency of several hundred Hertz. The axon connects to other neurons' dendrites via an adaptive connector called a *synapse*. There may be as many as 1 000 to 100 000 synapses for each neuron. Figure 3.1 presents some synapse configurations. When an electrical signal is transmitted from an axon into a synapse, it enters the synaptic cleft where special chemical neuro-transmitters are released. The transfer of information is done both chemically (99%) and electrically (1%). This active environment also changes the physical conditions of the synapse, which is part of the learning process. Action potentials take on either excitatory (positive) or inhibitory (negative) states. The degree of activity in the synapse is controlled by the strength and sign of the incoming action potential, the degree to which the neuro-transmitters are released into the synaptic left, which neuro-transmitters are present, and their relative concentrations. The signals travel only in *one* direction though - down the dendrite to the soma and out of the soma along the axon. The soma sums all the incoming signals *temporally* using a milliseconds order time constant. Depending on the magnitude and sign of the resulting summation, it may generate an excitatory post synaptic (EPSP) action potential if its internal threshold is exceeded.

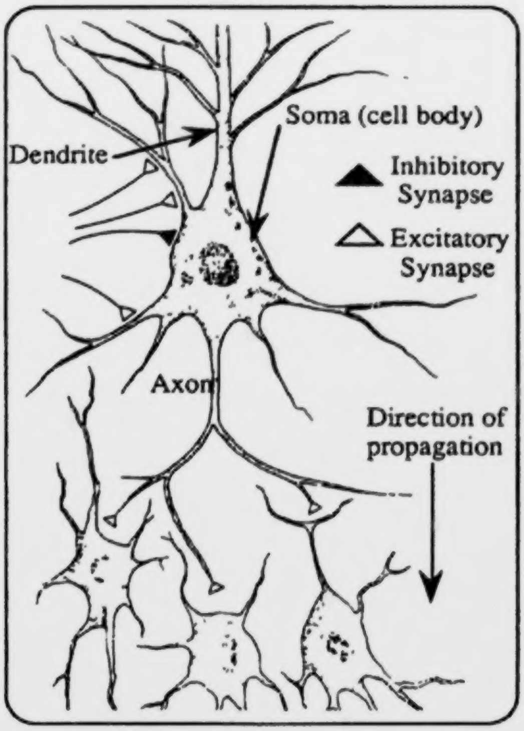

Fig. 3.1 Biological neuron and synapse descriptions (scanned) after [KaSc85].

Similarly, the neuron may invert the signal generating an inhibitory post synaptic potential (IPSP), or it may simply stay at rest due to insufficient stimulation. It also will remain at rest if the pulse trains do not arrive sufficiently close together or if the neurochemicals are spent due to continual firing. An excitatory action potential is shown in Fig. 3.2. The leading portion of an action potential is called the depolarization region and its trailing portion is called the hyperpolarization region. The depolarization peak may reach up to 50 mV while the hyperpolarization peak may dip to -70 mV. A resting neurons potential is approximately -60 mV partly because of the different sodium, calcium, and potassium ion concentration present at the cell membrane. The following list includes only some characteristics used by neurons to produce observed cognitive activity: (i) connections, (ii) various chemical strengths developed between connections, (iii) strength and sign of the output signal, (iv) threshold and time constant of the soma, (v) number of connections made to the soma, (vi) type of neuron, and (vii) the travel time of the signal along the dendrites and axon.

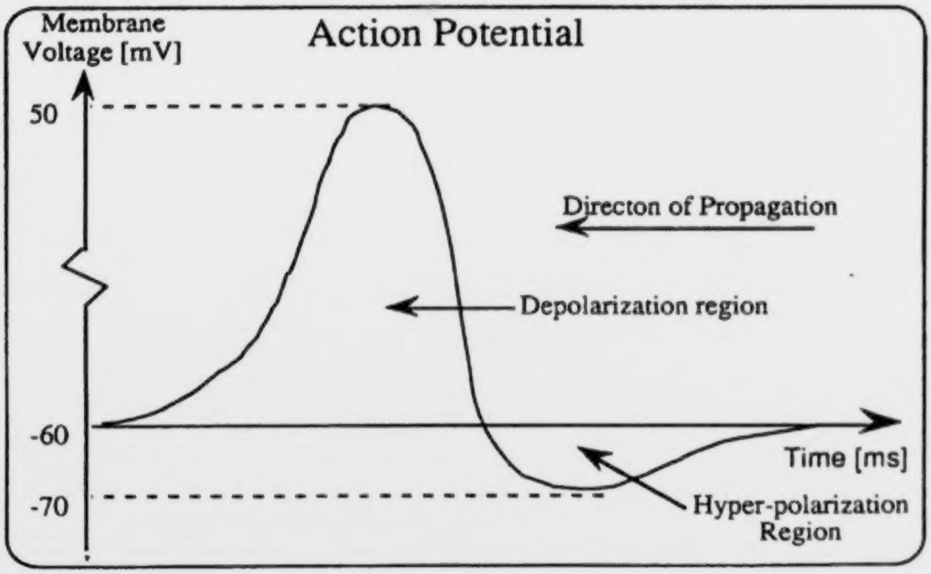

Fig. 3 2 An excitatory action potential.

The huge parallel network found in the brain enables cognitive functions, namely learning and memory. It is commonly believed that stimulus percolates through each layer creating increasingly higher levels of abstractions, which forms more complex ideas. Researchers have, where they could, implemented many properties of neurons when developing ANNs. Often due to limitations in knowledge, software, and hardware, some approximations and compromises were necessary. This has been the approach taken to developing ANNs and some motivation behind the development of temporal NNs. The NNs that are discussed attempt to incorporate more of the observable temporal characteristics present in biological neurons, which includes temporal integration, the data flow characteristics of the axons and dendrites, and the temporal time constant. Unfortunately, it is not known exactly how we perform temporal segmentation, yet some good models have been developed. Therefore, the first step toward this goal is to understand how temporal segmentation is modelled.

3.3 Temporal Decomposition

This discussion describes an application of temporal segmentation, the basic *temporal decomposition* model and the improvements made to it, and finally describes its role in temporal speech recognition. Speech can be considered a sequence of distinct articulatory gestures where each gesture produces an acoustic event that approximates a phonetic target. In natural speech, these overlapping gestures appear as phonemic transitions. Temporal decomposition of speech is based on the assumption that speech production, including coarticulation, is described by a *linear combination* of articulatory target positions given a suitable parametric representation. The speech parameter representation should reflect this linearity [Dijk89], i.e., each target function and associated target vector should relate to one acoustic event. This technique [Atal83] was originally intended for encoding speech at low bit rates but has led to a land slide of research in the area of neural network based temporal speech recognition by [NiFa89, BaMA89, MCCH89, DiMa89, Dijk89, BCDM88, and DBMC88].

3.3.1 Temporal Decomposition Model

The temporal decomposition (TD) model is motivated by the observation that sample-by-sample quantization of linear predictive coding (LPC) parameters is not very efficient. To encode speech at a constant sampling rate, the fastest occurring acoustic event is determined and the sampling rate is set to capture it accurately. Because *all* events do not require this maximum sampling rate much encoding redundancy is introduced [Atal83]. A redundancy reduction method was developed that separated acoustic events such that, ideally, only one target function represented an acoustic event instead of multiple ones. This *new* method, TD, encoded speech using a nonuniform sampling process. The location and size of each acoustic event was adaptively modified to minimize the amount of redundancy. Atal assumed that coarticulation could be modelled using a linear combination of underlying target functions. The model had problems though, since minor adjustments to the input parameters caused extreme prediction errors of the temporal event locations. Also, his method did not account for any articulatory or phonetic speech knowledge other than the facts concerning coarticulation and the dynamic and phonetic trajectories of speech. Improvements were necessary to make this approach effective for CSR.

Model Background

The temporal decomposition process is similar to linear predictive coding (LPC) [Pars86] in that the speech samples are represented by $y_i(n)$, and the predicted ones by $\tilde{y}_i(n)$ and, also much like LPC, is the TD fundamental relationship expressed by Eq. 3.1, where $\emptyset_k(n)$ is the *k*th target function and a_{ik} is the target vector.

$$\tilde{y}_i(n) = \sum_{k=1}^{m} a_{i,k}\emptyset_k(n),\ 1 \le n \le N,\ 1 \le i \le r, \tag{3.1}$$

The letter *m* corresponds to the maximum number of acoustic events in an utterance with the interval ranging from $n = 1$ to N. This value is based on Rabiner and Schafer's discussion, which states that approximately ten to fifteen acoustic events occur per second [RaSc78]. Equation 3.1 is represented in matrix format by Eq. 3.2 where $\widetilde{\mathbf{Y}}$ is an R x N matrix whose (i, n) element is $y_i(n)$, A is an R x M matrix whose (i, k) element is $a_{i,k}$, and Φ is an M x N matrix whose (k, n) element is $\emptyset_k(n)$. Unfortunately, both A (target vectors) and Φ (target functions) are unknown.

$$\widetilde{\mathbf{Y}} = \mathbf{A}\ \Phi \qquad (3.2)$$

The data used by the model begins with a parametric speech representation. Time ordering each $\emptyset_k(n)$ implies that each target function represents an acoustic event. These target functions are based on the use of the log area ratio parameters (LAR), which closely parallel the slowly varying speech parameters and exhibit the necessary high covariance that makes them ideally suitable for TD. At this point, a brief description of LAR parameters is given for background.

LAR Parameters

The LAR parameters are derived from the filter coefficients of LPC. Initially, the LAR parameters are based on area coefficients (A). Area coefficients represent the cross sections of the *I* successive sections of the vocal tract. The *A* parameters are related to the LPC reflection coefficients, *k*, as shown in Eq. 3.3,

$$A_i = A_{I+1} \frac{1 + k_i}{1 - k_i},\ A_{I+1} = 1,\ 1 \le i \le I \qquad (3.3)$$

Now the log area (LA) parameters are the logarithms of the *A* parameters, i.e., $\log A_i$, $1 \le i \le I$. The LAR parameters, g, are expressed in terms of the LPC reflection coefficients, which are then represented by the *LA* parameters, as shown in Eq. 3.4,

$$g_i = \log \frac{1 + k_i}{1 - k_i} = \log \frac{A_i}{A_{i+1}}, \; 1 \leq i \leq I \tag{3.4}$$

The LAR coefficients, which comprise the most commonly used TD parameter set, are almost linear to the reflection coefficients for a large range of data values. Others (Viswanathan et al., 1975) have shown that the LAR parameters provide an optimal set for quantization [VaMa89].

3.3.2 Determination of the Target Functions

The first step in temporal decomposition is to solve for the target functions. This involves applying three restrictions to the target functions:

(i) $\emptyset_k(n)$ can be nonzero only over a short period (compact in time).
(ii) At every instant of time, only a finite number of $\emptyset_k(n)$ can be nonzero.
(iii) The $\emptyset_k(n)$s must be represented as a linear combination of LAR parameters.

A target function, given in Eq. 3.5 and shown in Fig. 3.3, is a linear combination of speech samples $y_i(n)$, where the w_{ki}s are a set of weighting coefficients and the letter m is used to represent the number of speech samples. The w_{ki}s are chosen such that the

$$\emptyset_k(n) = \sum_{i=1}^{m} w_{k,i}\, y_i(n), \; 1 \leq k \leq p, \; 1 \leq n \leq N \tag{3.5}$$

first two target function restrictions are satisfied. This implies that only a finite set of w_{ki}s is nonzero. Graphically, this is interpreted as placing a small window over the matrix of speech parameters, ***Y***.

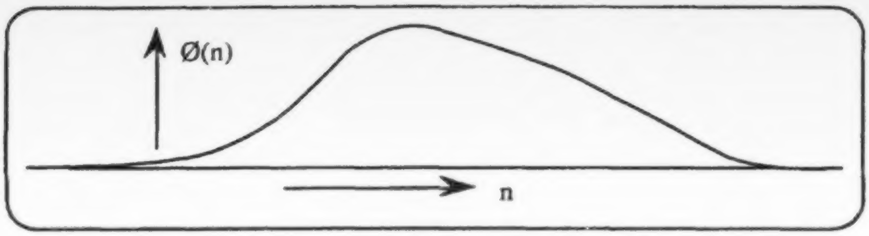

Fig. 3.3 An idealized sketch of a target function (after [Atal83]).

The first step in solving for the target functions is to perform singular valued decomposition (SVD) (Golub and Van Loan, 1983) as given in Eq. 3.6 where *U* is an *N* x *M* orthogonal matrix, *V* is an *M* x *M* orthogonal matrix and their columns contain singular vectors, and ***D*** is a diagonal matrix of singular values and its components are the square roots of the eigenvalues of Y^tY (superscript *t* refers to the transpose operation).

$$\mathbf{Y}^t = \mathbf{U\ D\ V}^t \tag{3.6}$$

The singular values limit how much variance is accounted for by the respective singular vectors. Three to five vectors usually account for about 95% of the variance. The speech parameters, $y_i(n)$, are expressed as a linear combination of the parameters, $u_i(n)$. Substituting this into Eq. 3.5 and then taking only the *s* most significant singular vectors results in an important reduction in solving for the $\emptyset_k(n)$ functions, as shown in Eq. 3.7,

$$\emptyset_k(n) = \sum_{i=1}^{m} b_{k,i}\, u_i(n) \tag{3.7}$$

where $u_i(n)$ is the element in the *n*th row and *i*th column of the matrix *U* and the b_{ki}s are a set of amplitude coefficients, which have to be appropriately *chosen*. To obtain the *optimal* target functions, replace the $\emptyset_k(n)$ given by Eq. 3.7 into Eq. 3.8. Equation 3.8 represents a distance metric, $\theta(\cdot)$, of $\emptyset_k(n)$ derived from the sample $n = n_c$ and is,

$$\theta(n_c)^2 = \left[\frac{\sum\limits_{n} \alpha(n)\, \varnothing_k^2(n)}{\sum\limits_{n} \varnothing_k^2(n)}\right] \tag{3.8}$$

where the sum over the index n extends over the duration of the speech segment and $\alpha(n)$ is a weighting factor. The optimal target function is found by minimizing this metric. It is easier to minimize $\ln \theta(\cdot)$, so that is the approach taken. The derivative of $\ln \theta(\cdot)$ is taken with respect to b_{ki} and set equal to zero. This math is used to find the eigenvalues, λ, of Eq. 3.9 with coefficient of $\boldsymbol{R}$, r_{ij},

$$Rb = \lambda b, \;\; r_{ij} = \sum_{n} \alpha(n)\, u_i(n)\, u_j(n) \tag{3.9}$$

Finally, by choosing the extreme eigenvalue from Eq. 3.9s solution and using it to decide the b_{ki}s, the target functions are easily solved using Eq. 3.7. These functions have been optimized with respect to selection but not with respect to shape and location, which is the purpose of the next section.

3.3.3 Determination of the Weighting Factor

Atal defines a measure of spread over the N frames with centre at n_c. The shapes of the target functions are dependent to a large degree on the weighting factor $\propto(n)$, where $\propto(n) = (n - n_c)^2$. Since $\propto(n)$ is quadratic in nature, it focuses strongly upon the centrally located target function. The target functions, which are intended to represent articulatory gestures, are not generally expected to be found at their central maximums. This also forces the functions to be very compact, which inhibits the search for longer acoustic events. A new rectangular weighting factor ($\propto(n) = 1$, $n_1 \leq n \leq n_2$ and zero elsewhere) iteratively adjusts both the length (n_1, n_2) and location of the target function to maximize the spread measure of $\theta(n)$, which is used to figure out the optimal $\varnothing_k(n)$. The window is modified until the current model, m_t, is equal to its predecessor m_{t-1}. This typically takes about three to four iterations. This is shown in Fig. 3.4 where the higher

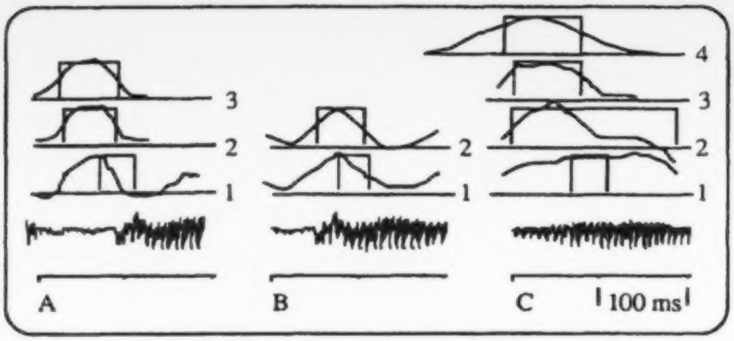

Fig. 3.4 Target functions resulting from various models for three segments of speech. The initial choice of the model is fixed in the centre of the window, and converge in successive iterations (after [VaMa89]).

number models represent the better approximations. An unreached target is modelled by the overlap of two or even three target functions. In order not to miss an acoustic event, Atal moves the window a fixed step size of ten ms. At this resolution though, many redundant identical functions are created. Atal's solution to reducing the number of redundant functions is to check for zero crossings. If a crossing occurs, then so too did a speech event. This approach does not guarantee a selection of only well-shaped and unique functions because it discards both redundant and relevant information. Secondly, because there is not always a rapid shift from one target function to the next a gap could occur because of this pruning procedure.

As a solution to this problem, instead of having a discrete window step size, the next window location is chosen where we *expect* to find a new $\emptyset_k(n)$. The optimum choice for this is at n_{right} (the right hand location of the currently known boundary). To prevent a very similar target function from being generated, the proposed target function is treated as a vector and the cosine is taken with respect to the current $\emptyset_k(n)$. If the cosine is greater than the threshold, 0.75, the proposed target function is discarded due to its similarity and the search is shifted $2*n_{right}$.

3.3.4 Determination of the Amplitude Coefficients

The amplitude coefficients used in scaling the various target functions to estimate the acoustic events are calculated by minimizing the mean squared error *E*, given in Eq. 3.10, and taking the partial derivative of the error with respect to the amplitude coefficients and setting the result to zero. The letter *m*, again represents the total number of speech events within the range of index *n* over which the sum is carried out, where the letter *n* represents the number of basis functions.

$$E = \sum_{n} \left[y_i(n) - \sum_{k=1}^{m} a_{i,k} \, \phi_k(n) \right]^2 \qquad (3.10)$$

Solving the system given in Eq. 3.11, yields a set of acoustic target vectors, each of which has a frame of ten (m=10) LAR coefficients. A further refinement was introduced by

$$\sum_{k=1}^{m} a_{i,k} \sum_{n} \phi_k(n) \, \phi_r(n) = \sum_{n} y_i(n) \, \phi_r(n), \; 1 \le r \le m, \; 1 \le i \le p \qquad (3.11)$$

[Atal83] to reduce the amount of ripples produced by the combination of amplitude coefficients and target functions as compared to the actual lobes in the acoustic events shown in Fig. 3.5. This refinement, given by Eq. 3.12, typically involved about four to five further iterations. The refinement also truncated portions of the lobes, which reduced the number of bits used to encode the event. These lobes occurred due to neighboring

$$\phi_r(n) = \frac{\left[\sum_{i=1}^{p} y_i(n) \, a_{i,r} - \sum_{k \ne r} \phi_k(n) \sum_{i=1}^{p} a_{i,k} \, a_{i,r} \right]}{\sum_{i=1}^{p} a_{i,r}^2} \qquad (3.12)$$

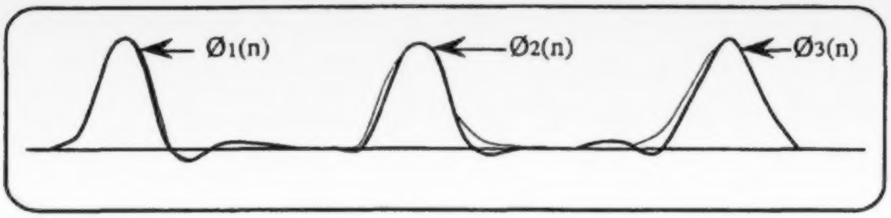

Fig. 3.5 Plots of the target functions obtained by temporal decomposition of three curves. The light curves are the actual target functions (superposed upon the improved target function estimations) (after [Atal83]).

speech events. For an undersized window, Atal selected some different target functions. Again, this did not guarantee the selection of only well-shaped functions. The new approach [VaMa89] determines both event boundaries independently. A check is also made to ensure that the target function does not lie entirely outside these boundaries. The new window adjustment results are shown in Fig. 3.6 where the targets numbered *one* are the *resulting* targets of Fig. 3.4 and the highest numbered targets show the *new* and final, optimally adjusted functions with the additional window length and location adjustments.

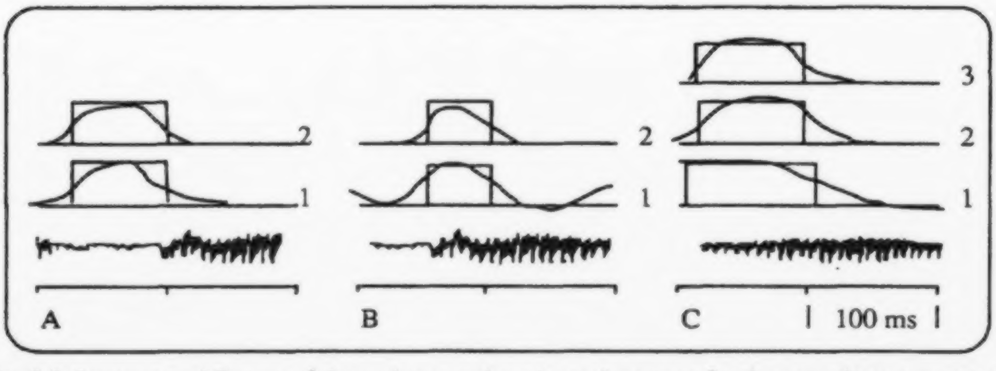

Fig. 3.6 Iterative modification of the analysis window size and position, for the same three segments of speech as above in Fig. 3 4 (after [VaMa89]).

The error criteria, *E*, given by Eq. 3.10 is checked until it falls below a threshold. In Fig. 3.5 the target function error prior to the refinement is shown as dark lobes with ripples. The light lines are the actual target functions. They were tested out on several sentences spoken both by male and female speakers. A resulting speech example with the target function estimates is shown in Fig. 3.7. There are 23 distinct target functions that overlap in time for the spoken sentence, 'Joe brought a young girl.' "As expected, the target functions for short transient sounds last over a short time interval while the target functions for relatively stationary vowel sounds lasted over a much longer time interval" [Atal83]. Although this technique was intended for encoding speech, it also provided an efficient approach to temporally decomposing continuous speech into acoustic temporal events. The improvements describe an approach where *all* the target functions are found, the redundant target functions are minimized, and the shape and location of the target function is optimized, thus requiring fewer target functions to represent the acoustic events.

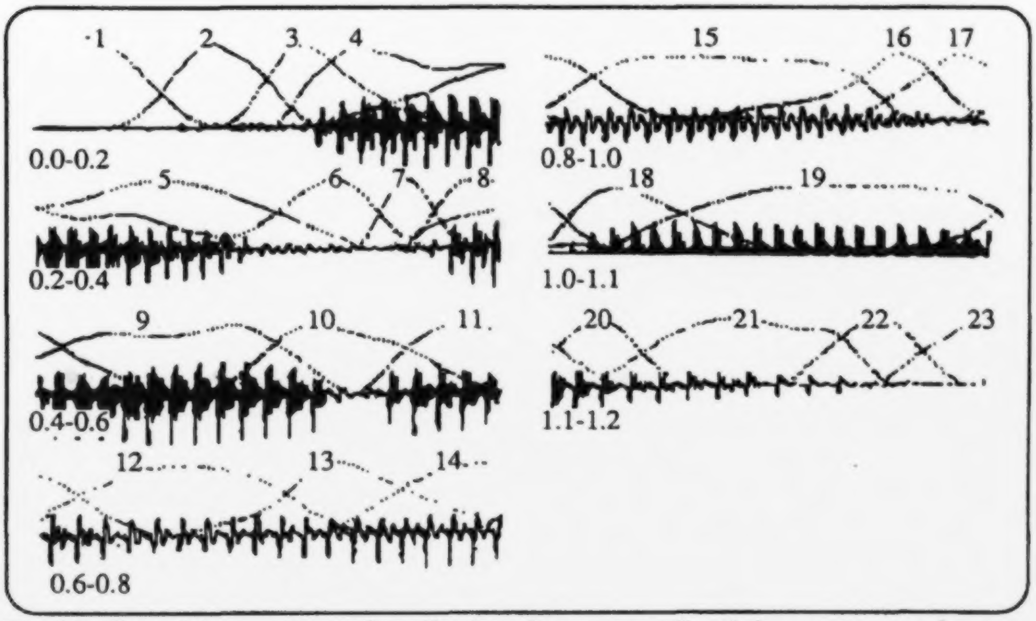

Fig. 3.7 Plots of the speech waveform 'Joe brought a young girl' and the various target functions determined by the temporal decomposition technique. The time intervals for the different segments are marked in the left margin in each case ({scanned} after [Atal83]).

3.4 Speech Segmentation Techniques

This section describes some improvements to speech segmentation in time and speech segmentation using parameters. As well, a brief description of past approaches to segmentation is provided.

Speech is not produced in discrete isolated units. Instead, it overlaps from one phoneme into the next and sometimes overlaps from one word into the next. Having an identical phoneme follow two different phonemes as in /da/ and /ga/, cause the common phoneme, /a/, to be pronounced or articulated differently. Thus, it is computationally intractable to attempt to learn all possible phoneme combinations. We would have to consider also dialect, rate of articulation, the language itself, and many other factors. "It is obviously impractical to do recognition of whole phrases because the number of possible phrases is too vast. Hence, the process must break the input stream into constituent parts, as people do" [Pars86]. Contemporary systems based on this idea would then have to recognize word boundaries, which evidently is very difficult to do if any degree of accuracy is desired since there is no reliable feature that can find where one phoneme (or word) begins and where it ends. Segmentation algorithms for speech recognition typically take amplitude as a starting point with energy minima and zero crossings. These have severe limitations with respect to many phonetic arrangements. Two classes have evolved though; the first class is segmentation in time, which cuts the waveform into nonoverlapping segments. The other class uses features of the waveform to decide boundaries, the results of which, may overlap.

3.4.1 Linearity and Invariance Conditions

The linearity and invariance assumptions (shown in Fig. 3.8), upon which many classical systems are based, emphasize the point concerning the ignorance of past speech recognition model segmentation philosophies.

Linearity Condition: If abstract unit *A* is before *B* then there is a segment of speech that is the physical form of *A* and it precedes the segment that is the physical form of *B*. They are like beads on a string; unrelated and discrete (segmentation in time).

Invariance Condition: Every segment *A* has a defining set of acoustic features $f_1^A, f_2^A, f_3^A, \cdots, f_N^A$ for some *N*, in which all features occur for every occurrence of *A*, and that a full set of features *never* occurs for any segment *B*, that is distinct from *A*, i.e., no intonation or coarticulation between units!

After some consideration most phonemicists would reject these conditions although many CSR systems use them. Chomsky and Miller offered one of many examples for which both conditions fail. For example, the words *writer* and *rider* phonemically differ in their fourth segment but differ phonetically in their second segment, not in their fourth.

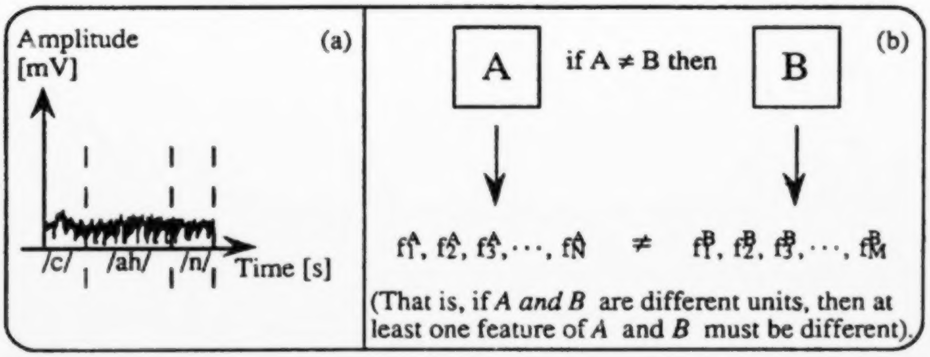
Amplitude
[mV]
(a)
/c/
/ah/
/n/
Time [s]
A
if A ≠ B then
B
(b)
$f_1^A, f_2^A, f_3^A, \cdots, f_N^A \neq f_1^B, f_2^B, f_3^B, \cdots, f_M^B$
(That is, if *A and B* are different units, then at least one feature of *A* and *B* must be different).

Fig. 3.8 Description of the linearity and invariance condition, which implicitly limit many recognition systems (after [Bris86]). (a) Shows an example of the linearity condition for the word /can/ involving nonoverlapping time segments while (b) describes the invariance condition involving unique sets of feature vectors.

"Speech performance involves continuous movement of the vocal apparatus; static postures are few, and pauses are left (sometimes) between intonation units, not between words and certainly not between allophones. The result is an acoustic signal in almost certain flux, whose steady state portions often do not correspond straightforwardly to phonemes" [Bris86]. Thus, these philosophies must be changed since further advancement cannot result from them, especially when considering continuous speech.

3.4.2 Segmentation According to Time

The speech signal may be segmented according to some phonetic analysis but if this higher level speech knowledge is ignored, speech can be discretely segmented according to time. Segmentation of speech by time slicing it into discrete segments has had little success. Here, all things are treated equally, though some signal differences are more important than others. Since we cannot even discriminate between even two *close* signals we must store all variations of each word that evidently leads to a physical storage problem. To reduce the effects causing these differences, the following modifications may be considered:

(i) Dynamic time warping (DTW) - This would reduce the timing differences between spoken cases of the same utterance by time aligning the utterance (that was spoken at a different rate from the template - refer to section 2.3) using localized compression and expansion of its components.

(ii) Fundamental frequency (F0) - A better DTW match results by imposing the F0 from the templates onto the unknown utterance. This reduces the effect of pitch variations of the same utterance spoken, perhaps, by different genders, or age groups..

Although this section is *supposed* to be focusing on *strict time segmentation*, it is difficult not to use other techniques to help in the segmentation process. Since segmentation according to time is unreliable, one should ask if there is some unambiguous cue that suggests when one sound begins and another ends? Unfortunately, the answer to this segmentation problem is, *'not really'*. "Segmentation is an errorful classification problem in itself . . ." [Waib89a]. A suitable model is needed that scans the time sequenced frames of speech, from which reliable acoustic low level cues are extracted and used to form word boundaries locations. This approach implies that these detectors must be invariant to time and location of the acoustic event and act temporally. These *'reliable acoustic low level cues'* are the next topic of interest.

3.4.3 Segmentation According to Parameters

There are several parameters available that characterize speech well and help to decide the segmentation boundaries. The first parameter is the fundamental frequency, F0, of voiced sounds. Caution should be used since voicing also occurs in other groups that are not voiced. Although voicing is rarely used *directly for segmentation purposes* it *is* used to fit *pitch contours*, which are determined using the intonation contours and stress value characteristic to English. The contours provide *islands of reliability* upon which segmentation boundaries are established. The second parameter that may be used is intensity. Intensity is observed as the sharp time dependent variations in the speech signal. This is a very important parameter since it often suggests changing speech sounds. For instance, a low intensity may suggest a pause, weak fricative, or a stop closure. Intensity may be measured using amplitude (energy) and found through filtering. The first three resonating frequencies of the vocal tract (formants) can be used to classify vowel and sonorants, and formant trajectories may be used to classify diphthong boundaries. The gross spectral shape has been used as a rough estimate of boundaries for fricatives or the onset of plosive releases. The zero crossing frequency can be used as a boundary detection feature. In cases involving nasalization, sonorant and fricatives, voiced and unvoiced sections, abrupt changes occur, which again may prove useful in boundary detection.

In summary, although there are many parameters that may be used to detect boundaries and because there is no single speech waveform parameter that is *always* reliable, many different parameters must be used together to identify speech boundaries. Even then, they do not always perform well since the boundaries found by the parameters do not always agree! It appears that segmentation of speech will *always* have a built-in error factor with respect to continuous speech.

3.6 Summary

This chapter discussed some features that are used in temporal speech recognition. A brief look at temporal processing was given from the biological perspective and then of its use as a practical feature. A detailed description followed on temporal decomposition, which modelled coarticulation and varying utterance lengths extremely well. Finally some common segmentation approaches were examined, both from a time and parameter domain. In both cases it was determined that segmentation would always have a built-in error margin because there was no *one* parameter or approach that could be used as a reliable basis for segmentation. Many parameters were necessary to cover the domain of speech categories and the parameter boundary estimations usually did not agree closely. To help this end, it is useful to enhance or modify the speech signal both before and/or after segmentation, which is the topic of the following chapter.

IV
MODIFICATIONS AND ENHANCEMENTS TO THE SPEECH SIGNAL

4.1 Introduction

The purpose of this chapter is to consider methods of modifying or enhancing a speech signal that is temporally segmented to improve recognition. The main objective of speech enhancement is to improve one or more perceptual aspects of the speech signal such as total quality, intelligibility, or degree of listener fatigue. Often systems perform poorly in noisy or adverse environments. This makes the segmentation and recognition process more difficult. These effects are reduced by applying nonstochastic methods to the speech segments. For instance, the coarticulation effect of *temporally* segmented speech is reduced because the temporal decomposition process finds the *overlapping boundaries*.

Prosodic systems (stress and intonation) are investigated that use temporally segmented speech samples to help recognition. The philosophy is *not* to remove properties of the speech waveform through modifications, but to understand first its meaning and operation, and then to incorporate advantageously each into the recognition process. In other words, turn something that may be *causing a problem* into a system *asset*, since each prosodic system carries invaluable information about the speech.

The next set of short discussions describes a noise cancellation approach and some prosodic systems that are useful for either improving the recognition process or helping in the temporal segmentation process. Sometimes, certain prosodic systems do both.

4.2 Noise Reduction

In the presence of noise, speech recognition performance degrades rapidly. This performance drop is often because the system is trained on noiseless speech. Intuitively it appears that the network should, therefore, train using noise corrupted speech that approximates the recognizer environment. The system will fail though if it then operates in the absence of noise!

The noise in question is called *acoustic ambient* noise. This noise is usually additive, which means that the recorded signal is composed of a message added (linearly) to the ambient noise. Ambient noise can manifest itself as that noise generated from devices such as printers, computers, telephones, fans, and as background conversation. The spectrum of ambient noise, as found in a car, is not flat; it is higher in magnitude for low frequencies and lower at high frequencies. Some noise types are nonlinear, which besides the linear ones, represent an accurate representation of the noise that accompanies a message signal. It is encouraging to note that NNs are very well suited to deal with both linear and nonlinear relationships. This NN ability may very accurately extract the noise from the message signal. No evidence supporting this presumption has been put forward yet although [KaSo91] have done some early work to this end. They investigated a vector quantization (VQ) mapping on the decoding side of a communication system where noisy samples are mapped back to noiseless samples. The codebook only had 16 words in it but results were encouraging.

Another approach that helps in reducing the effects of noise is *multistyle* training. This involves using a training set that contains copies of the desired library, each of which has been exposed to the various types of noise environments. The other alternative, which may be used with the above one, is to employ enhancement methods that suppress the noise prior to recognition. One such approach is adaptive noise cancelling.

Adaptive Noise Cancelling

This technique "adaptively adjusts the filter coefficients so that subtracting the filtered (noise) reference input from the primary (speech plus noise) input leads to an input signal with minimum energy" [Juan91]. The technique requires correlation of the noise corrupted signal and the noise signal. Unfortunately it is difficult to get both high correlation and sufficient separation, and prevent the noise reference signal from corruption of the speaker's voice. To get a high correlation (>90%) the speaker's signal and noise reference pickup must be separated by five cm. At this distance it is impossible *not* to corrupt the noise reference. To isolate the noise reference, the noise pickup must be at least 50 cm. away; this restriction does not provide high correlation. At this distance only periodic noise such as engine rpm noise can be successfully removed. A two-input adaptive filter is shown in Fig. 4.1.

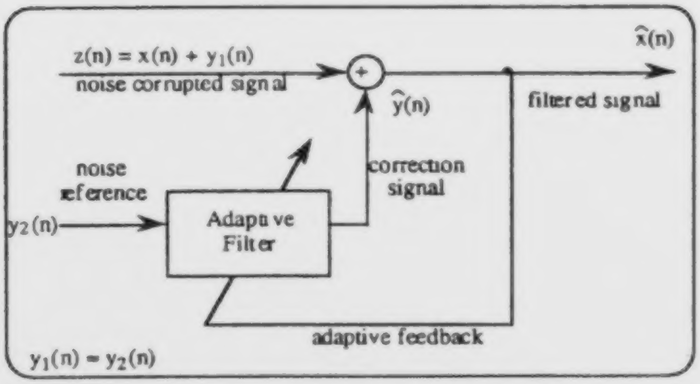

Fig. 4.1 Noise cancelling approach (after [Juan91]).

To improve noise cancellation others have investigated modelling the noise characteristic using SNR and/or noise power to obtain improved spectral models of speech from noise corrupted signals. This type of modelling eliminates the need for the actual noise reference, which is a limiting factor in the adaptive filter approach.

4.3 Intonation Adjustment

"The use of pitch to distinguish whole utterances without interfering with the shape of the component word is know as *intonation*" or more simply as, "modulation of the voice in speaking" [YaHu87]. Understanding intonation as the *flow of a sentence* can help in determining word boundaries because intonation is usually isolated to a single word. The difference in intonation of the word {*No.*} and {*No?*} contains pitch (intone) information that a simple recognizer ignores. Intonation, or in general pitch, can help first locate boundaries in the segmenting process and later provide insight to the word's *meaning*. This would enhance both the reliability and success of recognition.

The use of intonation contours in a recognition system can provide contextual cues over a word or phrase. In a more complex system, recognition of phrases by using sentence context and intonation may prove useful. This is especially true in situations where an utterance in the sentence is difficult to recognize. Here, intonation contours can improve the chance of identifying the utterance. Recognition of intonation contours involves three fundamental steps: (i) extract the pitch contour of the speech segment using a pitch tracker, (ii) process this contour by removing the noise while identifying robust voiced regions and smoothing irrelevant signal anomalies, and (iii) take the resulting parameters from step (ii) and categorize them. There are three type of intonation categories: (i) statement, (ii) interrogative, and (iii) exclamatory. Intonation processing details and algorithms are in [Waib88]. Granted, this system would have to be *trained* to deal with context and intonation, but at the phrase and sentence level, this may prove to be a useful and rewarding venture.

4.4 Stress Adjustment

Stress is an important notion for prosodic analysis. The purpose of stress adjustment is to provide compensation for spectral distortion that is caused by extraordinary speaking effort, which, in turn, is due to the talker's reaction to ambient conditions. No current analytical approaches to modelling stress are known so only good heuristic approaches are discussed.

Stress is the degree of effort exhibited during the production of a syllable. This sometimes correlates with the loudest part of a word. This was later determined uncertain since certain vowels produced with the *same* effort produced *different* amplitudes due to vocal parameters, and therefore, were characterized by different stress levels. Many algorithms can detect and classify stress by locating the stressed syllable in isolated words. Other methods developed by [Lea73a] obtain stress information from continuous speech, but the resulting performance is limited. Stress should not be considered in a binary fashion but more as a multilevel parameter. Stressed syllables are usually the best spoken syllable in a word and thus, provide a strong phonetic presence and reliability for the identity [Lea80, Lea73b, KlSt72]. These researchers have shown that a strong correlation exists between correct recognition and a recognizer which uses the stressed syllable in a segment for recognition. "Stressed syllables in a large English dictionary carry more acoustically discriminatory information than unstressed syllables, and therefore, provide not only more acoustic reliability, but more discriminatory information content" [Waib88]. The content words of a continuous sentence carry semantic meaning and are usually stressed while the grammatically predictable functional words such as, *the, at, it, or, of,* etc., are usually unstressed. Word-level stress is called *free-stress* in English because the stressed syllable can occur anywhere in the word. In other languages (French, Polish) stress is bound to a particular syllable and can be used to help a boundary detector. At higher construct levels, above phonemic and word level, there is contrastive and emphatic stress. *Contrastive* stress is used to contrast or negate two different items in a sentence while *emphatic* stress is used to distinguish a sentence from its negation. English stressed patterns are used as a lexical constraint to help identify a word given the word context and sentence rhythm pattern. For instance, if the last syllable of the previous word and its context is known, and if the stressed syllable in the current word is also known, then this information can help build a reliable word candidate subset.

To reduce English stress, identify the stressed syllable and then reduce its amplitude to some average in keeping with the localized region about this syllable. The amplitude is reduced since this is commonly how researchers modify stress. Thus, by locating the stressed syllable stress can be controlled. Stress is mainly due to variations in the fundamental frequency although F0 (pitch) produced far greater *perceptual* effects on stress than either intensity or duration [Waib88]. Yet, although most studies concluded that F0 is most responsible for perceptual changes in stress, amplitude is still the most commonly used parameter to affect stress, i.e., pitch is sufficient, but an unnecessary condition for distinguishing stressed vs. unstressed syllables.

One approach that uses stress to help in recognition is that of (Roe 1987 [Juan91]). The system uses vector quantization and DTW to produce *stress-free* speech. The process begins by first vector quantizing the spectral unstressed pattern and then replacing it by the closest spectral unstressed prototype of the VQ codebook. A reference template thus has a sequence of VQ spectral prototypes. "To adapt the *stress-free* clean spectral prototypes to the stressed speech, each (known) stressed speech utterance is time-aligned, without spectral quantization, to the correct reference template. The autocorrelation vectors of the stressed speech frames are then grouped according to the corresponding VQ indices in the reference templates and averaged to yield the *stress-compensated* prototypes" [Juan91].

This approach reduced the number of recognition errors by two-thirds when used in a speaker dependent isolated digit recognition system. This approach can be used in a NN system by training the net with stressed speech at the input and stress compensated prototypes at the output as determined a priori by the VQ codebook approach. The drawback here is that the stressed speech must first be recognized. In this research the recognition is done using DTW that has severe limitations in the continuous recognition domain as described in chapter III. After recognition, the unstressed autocorrelated version is aligned with the stressed VQ prototype and an averaging process is applied to the components. Recognizing the temporally decomposed segments may not prove to be easy. Nor is this approach *using* stress in recognition; on the contrary it is attempting to remove it, thus reducing the amount of effective signal information. It is useful to know where, when, and how stress is used in utterances and then train another network to help the primary recognition network. This is analogous to a sonar operator using a NN sonar signal recognizer to help in *reassurance* of a particular human sonar recognition decision.

To summarize, stress was defined as the degree of effort applied to a particular word syllable. English is characterized by *free-stress*, which implies that the stressed syllable can appear anywhere in a word. Amplitude is usually applied to measure stress although perceptual studies found F0 the most effective measure. To enhance a recognition system's worthiness (at the sentence level) stress can help by locating boundaries since the stressed syllable provides reliable acoustic information. To reduce the stressed syllables effect at the word level, reduce that syllable's amplitude. Finally, "if a phonetic transcription of the stressed syllable in a word was only given (with wild cards for the unstressed syllables), a substantially smaller set of words will remain as legal candidates than if only the phonetic transcription for an unstressed syllable was given" [Waib88]. This is to say, stressed syllables should be used for recognition purposes because they are more reliable in English and they provide more discriminatory information than unstressed syllables. Finally, a recognition system was described where stressed utterances were identified and the unstressed matches were used to compensate the stressed version to yield a *stress-free* version.

4.5 Summary

In this chapter, properties of the speech waveform were described, which were used to enhance or modify speech segments. These enhancements are intended to increase recognition. The first enhancement involved intonation, which relates closely to pitch. Intonation provides additional cues about the context of the word based on the intonation contour, which may be useful for recognition. The second enhancement involved stress, which measures the effort used in producing a syllable. In English, free-stress suggests that the stressed syllable may appear anywhere in a word. Although it may appear anywhere, it is usually in a known position in which a certain context and speech flow exist. If this fact was incorporated into a recognition system, then improved recognition may result.

V

CONTINUOUS SPEECH RECOGNITION USING NEURAL NETWORKS

5.1 Introduction

The ultimate goal of computer automated speech recognition is for a computer to master speech recognition in the same effortless manner as us. This chapter investigates temporal neural network approaches that brings us toward this end. The chapter has three main areas that focus strongly upon temporal speech recognition. These areas are: (i) Time delay neural networks, (ii) Temporal backpropagation, and (iii) Hierarchical neural networks. Each area initially describes the philosophy of the network approach and then provides some instances to demonstrate their abilities, and describe both their architectures and techniques. This also includes advantages and limitations of each instance, which are discussed in light of continuous and temporal speech recognition. This chapter provides a very good indicator of the recent progress and achievements in temporal and continuous speech recognition using neural networks.

Neural networks are not a newcomer to the area of continuous speech recognition. They are "good vehicles for automatic speech recognition not simply because they provide nonlinear pattern recognition but because their architecture allows the incorporation and exploitation of existing knowledge about speech" [Huck90]. Most recognizers elude the topic of *continuous* speech recognition (CSR) due to the added problems. Classical non-neural network based CSR systems including the TANGORA dictation system, the SPHINX [LHHM89] speaker independent system and KEAL's [MBML89] speaker dependent system have limited capabilities because they do not use ***enough*** speech knowledge, nor do they use it in the manner necessary to simplify connected speech recognition. Although these systems are based on decades of solid theory concerning speech and language these "contemporary computational architectures place restrictions on what can be effectively exploited" [Huck90]. We know that *these* problems affect the accuracy of recognition but the present architectures do not reflect this knowledge! The specific problems and their solutions are discussed throughout the chapter, suffice to say that systems that attempt to manage coarticulation, temporal variability, context sensitivity, the speaking environment, among other variables will more often succeed.

5.2 Time Delay Neural Networks

In many NNs, there is no time dependent relationships imposed by the network between patterns. Patterns are those components (raw or transformed) of the speech waveform that are used to represent the speech. Often, this approach to pattern representation causes scaling problems. Since one pattern is being used to train the network, both the rate of articulation and any timing variations that the pattern exhibits are learned. But future productions of that speech pattern will unlikely have the same traits. This leaves a large responsibility on the network to compensate through generalization. Besides this problem, are the problems of learning the varying lengths of phonemes, words, and phrases, and learning the different temporal characteristics of these classes. In these cases the typical *one-shot* network would simply be overburdened and thus, provide unsatisfactory results. Instead, a network that maintains temporal context and invariance to time shifts is needed. The problem of temporal recognition is dealt with by *time delay neural networks* (TDNN) and scaling is dealt with by the *modularity and incremental designs* of these architectures. "Networks trained to perform a smaller task may not produce outputs that are useful for solving a more complex task, but the knowledge and internal abstractions developed may indeed be valuable" [Waib89a]. This philosophy is based on a model of human learning and should be applied to connectionist systems. The temporal relationships between successive patterns are established in one of three ways; first using delay links, and/or second using recurrent links and finally in the architectural design and connectivity of the network itself. "A set of trainable connections with different time delays enables the networks to discover temporal relationships between acoustic-phonetic events" [BoWa91]. The hidden units are most commonly used to form the necessary abstractions used in recognition and, although they are useful for many problems, they are "not easily applicable to decoding of human speech" [Waib89a]. The models to be described use many of these approaches to achieve temporal recognition. Each excels in a particular aspect, which is identified and described.

5.2.1 Temporal Flow Model

The first architecture to be investigated is called the *Temporal Flow Model* (TFM). This model appears to possess properties appropriate for recognition of time varying signals including speech [Watr90]. These networks are trained using gradient descent methods and use a specialized Newton iterative nonlinear optimization process to reduce the mean squared error (MSE) between the desired output and the network output. Various tests are performed by Watrous to evaluate the networks, some of which are included to show the TFM temporal characteristics. A strong point of this model is its ability to learn time-shift invariance to patterns.

Since speech communication is an inherently *temporal* process with time being the independent variable that orders the process of perception and production, it forces us to realize that "the representation of time in a connectionist model designed to be used for speech recognition is a crucial issue. The temporal processing characteristics of the TFM arise from the use of *delay links* and the possibility of *recurrent* connections" [Watr90].

The recurrent connections are used to decode sequential events [Jord86]. The feedback provided by the recurrent links helps the temporal integration process. It also may help in controlling the duration of the temporal sequences with a time constant [Watr90] and [YZTK89], and for smoothing and differentiation, which are used to detect utterance onsets and lengths. By including temporal propagation delays and recurrent links, besides the standard neurons, many robust and dynamic connectionist models are realized that may be well suited for recognition of temporally segmented continuous speech at various grammatical levels. Using delay links make this implementation more biologically plausible. First, the various degrees of temporal connectivity are expressed by the number of delay links used between consecutive layers. Therefore, relationships between events in time are expressed as integer distances between naturally ordered samples. Second, delay links help to develop time-shift invariance to patterns. "An acoustic event will be invariant over delays in arrival time to the network. It is important to note that specifying a Gaussian target function for the utterance forced the network to learn a dynamic response to the data" [Watr90]. This is especially important since speech perception is independent of the absolute time reference of the speech, and therefore, should be modelled into any connectionist speech recognition network. Hinton has stated that network *learning* of time-shift invariance to input is extremely difficult, so representing it directly in the model is a significant advantage.

TFM Description

The TFM uses the standard sigmoid function, Eq. 5.1, in limiting the output, $O_j(n)$, and a modified activation function, Eq. 5.2, to incorporate the propagation delay links, $P_j(n)$. The letter d represents the propagation delay along a link, and $w_{i,j,d}$ represents the weight connecting unit u_i to unit u_j. The model notation, which is augmented from

$$O_j(n) = \frac{1}{1 + e^{-P_j(n) + \theta_j}} \tag{5.1}$$

$$P_j(n) = \sum_{i,d} w_{i,j,d}\, O_i(n-d) \tag{5.2}$$

the standard one, describes the number of input, hidden, and output neurons (eg. 8-3-3) and additionally includes the number of propagation delays at each level (eg. $8\text{-}3_2\text{-}3_3$). This example shows that there are eight input neurons fully connected to three hidden neurons that have two-level propagation delay links for each connection between the input and hidden layer. Similarly, there are three output neurons with three-level propagation delay links for each connection between the hidden and output layer. This method does not describe, yet, the time magnitude of each delay nor does it mention the inclusion of bias neurons at each level. An explanation of these details will follow.

Model Training

Training was performed by minimizing the MSE between the network output and the desired output. This approach, like BP, uses gradient descent techniques, which take small steps in the negative direction of the weight gradient with respect to the partitioned error to reduce the global error. The step size is controlled by the learning rate. Sometimes the learning rate and other control parameters are monitored and dynamically changed during training, which reduces the training period. Watrous uses an enhanced iterative gradient descent technique developed by Brogden, Fletcher, Goldfarb, and Shanno (BFGS). This algorithm is a "quasi-Newton method which employs an iteratively approximated inverse of the second derivative matrix to orient the search direction toward the minimum of the function" [Watr90]. BFGS is supposed to have higher convergence rates than first order gradient descent techniques. Another second order gradient descent technique that is less computationally intense is that of [Kuhn87] and may be more advantageous for CSR.

Experiments

Two recognition experiments are described for particular speech categories. Other experiments were conducted by Watrous. The first experiment investigated the problem of *place*, which discriminated the accuracy of the place of articulation of voiced stop consonants in CV words followed by the vowels [i, a, u]. The second experiment involved *manner*, which investigated the manner of articulation of the voiced alveolars in initial position of CV words for the three vowels [i, a, u]. The consonants were [n, d, l, z, d_3]. These experiments used one speaker for both training and testing and were conducted in an untreated acoustical room.

Data Preprocessing

The analog waveform was initially preamplified, low pass filtered ($f_{cutoff} = 8$ kHz), and digitized at 16 kHz using twelve bit sample resolution. The digital speech data was passed through sixteen bandpass filter banks that were sampled at five ms. The linear band range of the filter banks extended from 300 Hz to 1 kHz. These outputs were full wave rectified, smoothed, logarithmically compressed, and sampled at 2.5 ms using eight bit resolution. This data was smoothed using a triangular wave function and down sampled to five ms. This data was automatically segmented to extract the desired speech portions.

Place Experiment

This experiment involved the *place* of CV words followed by the vowels [i, a, u]. An average accuracy of 98% was reported. Results of 98.5% were reported for CSR (Watrous et al.; 1987; 1988). The architecture was based on a set of spectral-temporal feature detectors that differed in resolution and were ordered in a hierarchical fashion from low level statistic observations (input layer filter bank outputs) to high level complex phonetic structures (output layer class discriminations). The input layer consisted of sixteen neurons that were fed to the sixteen processed outputs of the filter banks at five ms intervals. The next highest layer, the *patches*, consisted of seven groups of three neurons. Each patch was connected to four input neurons and overlapped in the adjacent two input neurons claimed by each patch (see Fig. 5.1). The purpose of using three neurons per patch was to form sufficient multiple short time features over *local* frequencies. Delay links of one through four were used. "These delays shifted the centre of the four sample patch unit by two samples to give an overlap in the time dimension" [Watr90]. At the next layer, *strip* units formed features of the spectral patch responses over time and were implemented using delay links of one through nine, odd intervals only. Each of the seven strip groups had three neurons. A strip group was fully interconnected to only a single patch group, besides the delay connections. The strip units monitored 60 ms of output from the patch units using their delays. The architecture and delay link profile of this experiment is shown in Figs. 5.1 and 5.2, respectively. The network was initialized with random link values ranging between ± 1 and trained using the BFGS algorithm.

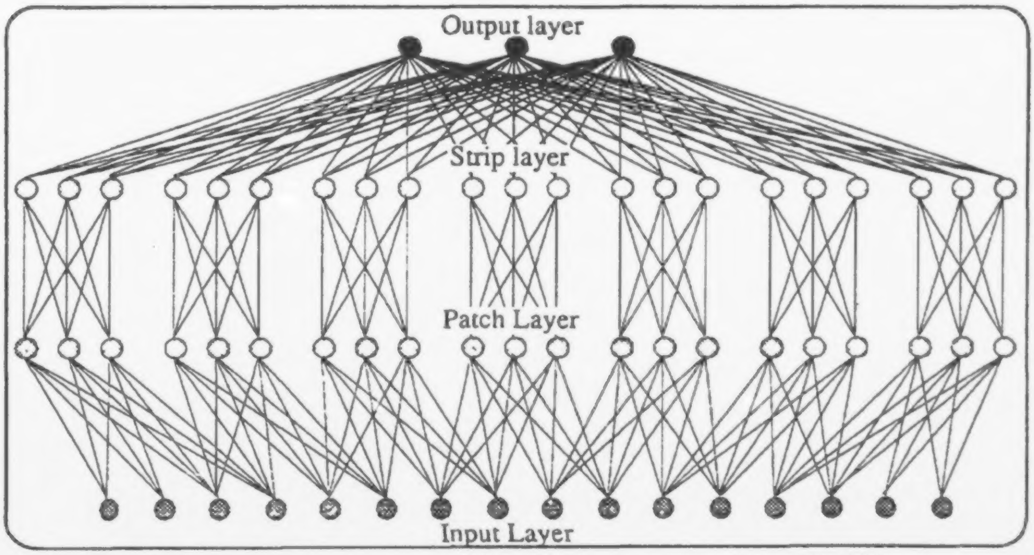

Fig. 5.1 Architectural overview of the TFM *place* experiment (after [Watr90]).

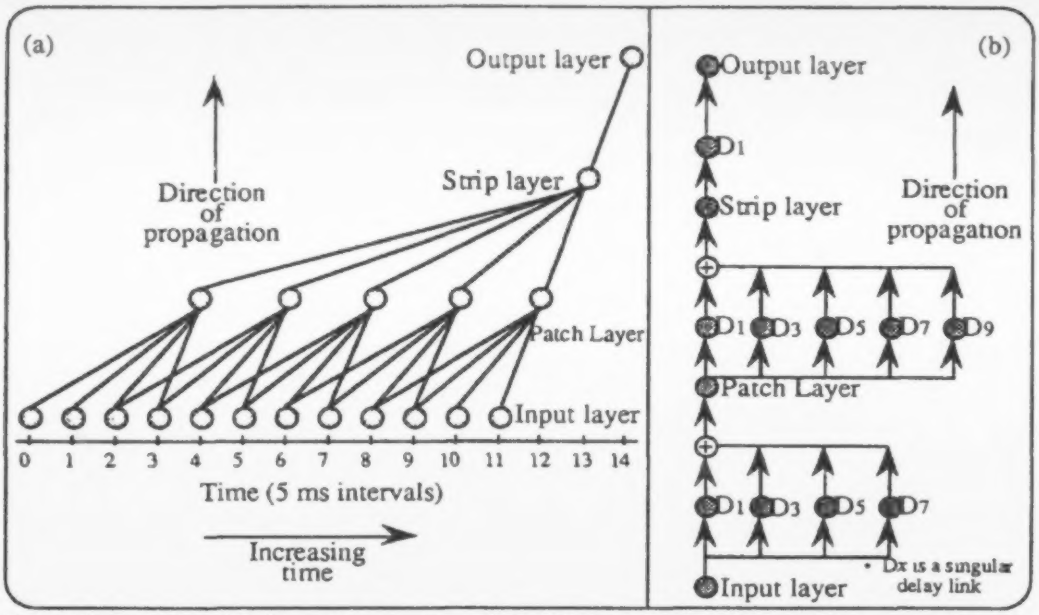

Fig. 5.2 Delay link profile for *place* experiment of the TFM. (a) Depicts the time delay propagation with a time reference as seen from *one* input neuron and its effect throughout the network (b) Depicts the time delay structure without a time reference (after [Watr90]).

Network Analysis

The response of the output units, which were trained for thirty repetitions is shown in Fig. 5.3 for some training data. Each output unit showed a positive response to the various CV data. Watrous found this perplexing given that the strip outputs were either flat or sloped and those of the patch units were almost all entirely flat across both vowel and the consonant. After further examination though, Watrous discovered that "the output unit responses were a function of the temporal alignment of the transitions of the *constituent* [I think Watrous meant to say *consecutive*] strips. The overlap regions between transitions caused net positive or negative pulses, depending on the polarity and sequence of the transitions" [Watr90]. Various discriminatory mechanisms were discovered at work within the network weight space, which with class boundaries were formed.

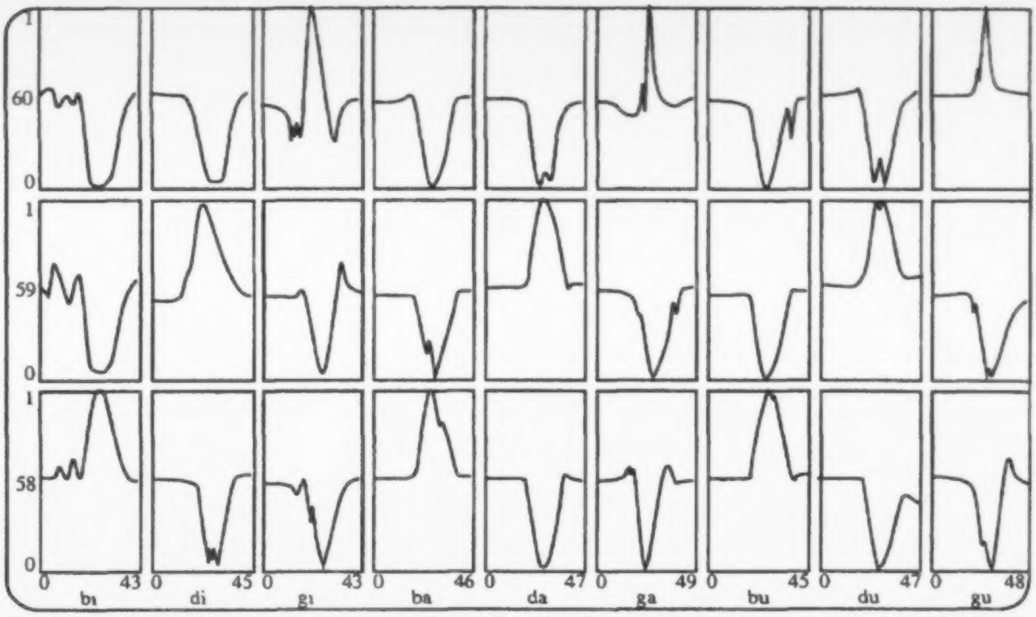

Fig. 5.3 Response of [b, d, g] output units (units 58 - 60) of place network were optimized on 30 training runs yielding an MSE of 0.3%. Display has been normalized for segment durations, which are listed as the number of five ms samples (after [Watr90]).

Many discriminations were largely based on frequency components and their combinations. This is reasonable given that the inputs were frequencies and, this is an efficient method of phonemic separation. Essentially the network had learned to respond to the initial stop consonant while providing as little output as possible to the silence preluding the consonant and the vowel following the consonant. The network did this by mapping the state of the strip units for the silence and the vowel onset to the same output state. The state of the strip units during the consonant release was mapped to the appropriate output unit. The network rejected the common parts of the pattern, silence and vowel onset, while extracting the relevant component, the consonant. This is a significant achievement considering the frequency content of a consonant is very broad and short in duration, especially for bursts. This meant that the discriminatory response involved a path through the weight space of the strip units beginning and ending at the same state. "This result was a dynamically balanced solution, which is a necessary step toward recognizing continuous speech" [Watr90]. Watrous says that a significant advantage of the connectionist approach is its ability to induce acoustic phonetic discriminations that are context variant to the extent required by the data. Further, this connectionist approach does not require prior segmentation of the data to isolate the interval of release burst because it simultaneously segments and classifies the pattern. The discriminatory features used in recognition were learned by the network using a statistical automated optimization process based on the speech data and it did not require external assistance in development. These cues may even be hidden or unknown to the individual when developing the speech data set.

Manner Experiment

The second experiment investigated the *manner* of voiced alveolars for the consonants [n, d, l, z, d_3] and vowels [i, a, u]. This network design involved an initial hidden layer consisting of sixteen neurons. The first thirteen *spectral slope feature* neurons were connected to four input neurons. An overlap of three input neurons existed between each spectral slope feature neuron. The last three hidden layer neurons were called *band energy* neurons. The combined distribution of the three bands captured all sixteen input neurons' activations. The distribution was five-six-five and no overlap existed between the claimed input neurons. The purpose of the band energy neurons was to extract pitch information across a phoneme or word. The twelve neurons in the next hidden layer were fully interconnected to the spectral slope feature layer and used delay link time values of one thru seven, odd values. This hidden layer formed compound spectral features based on the primary spectral features. Each hidden neuron spanned 30 ms of spectral slope feature activation. The five self-recurrent neurons in the output layer were fully interconnected to the previous hidden layer and used delay link values of one, five and nine. The output layer spanned a window of 40 ms of hidden layer activation. The architecture is shown in Fig. 5.4 and the delay profile is shown in Fig. 5.5.

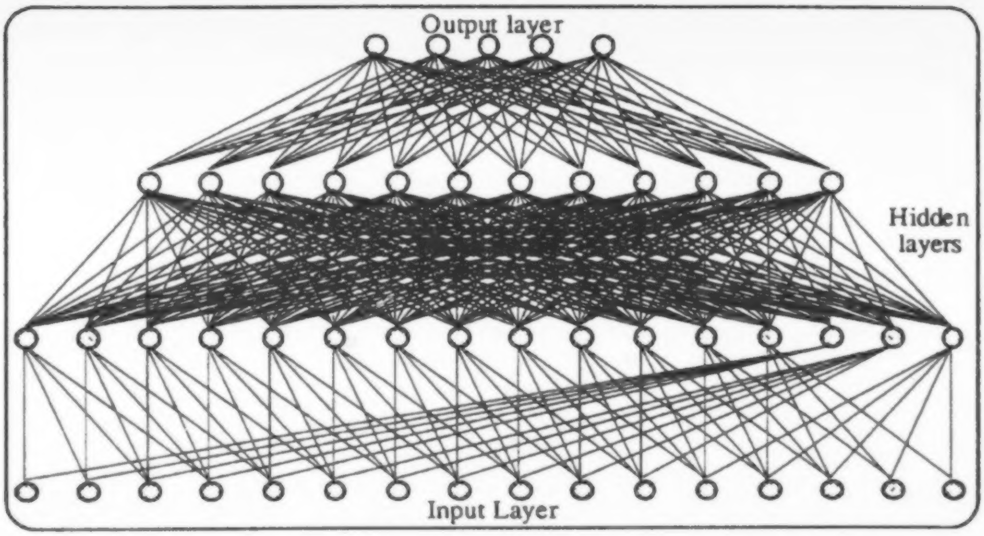

Fig. 5.4 Architectural overview of the TFM *manner* experiment (after [Watr90]).

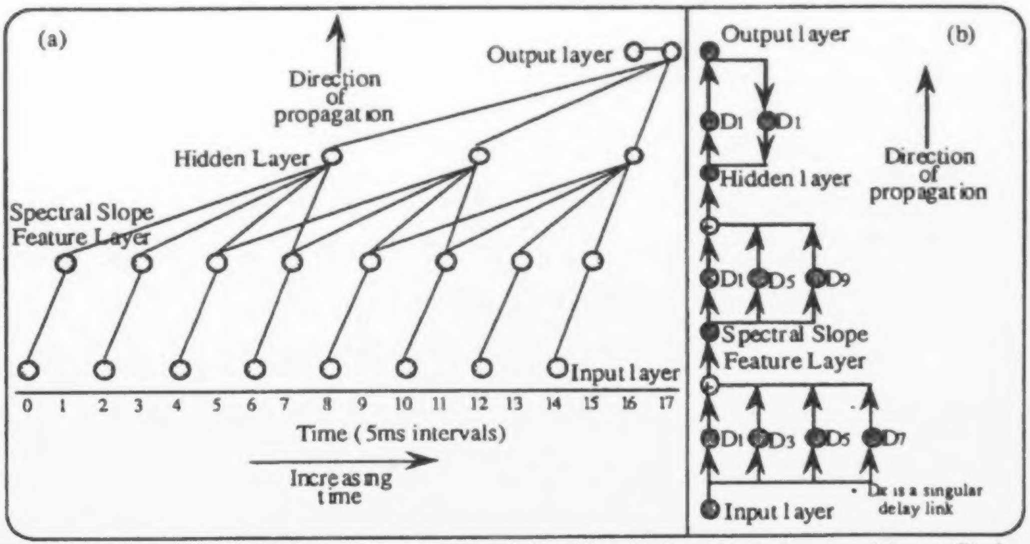

Fig. 5.5 Delay link profile for *manner* experiment of the TFM. (a) Depicts the dynamic delay profile for a single input neuron over time. (b) Depicts the static delay profile of the network.

Network Analysis

The phonemes [l, n, z, d_3] were identified based on one or more of the hidden units shown in Fig. 5.6. There was still no single hidden unit for which [d] was uniquely identifiable. "Analysis of the output units revealed that the classification of [d] depended on the temporal alignment of the transition edges of *two sets* of hidden units" [Watr90]. The *difference* in activation levels between the two sets of hidden units was responsible for isolating the release of the [d].

Since the pattern of the weights to the primary feature units did not change greatly, it is expected that "the feature inputs could have been fixed rather than learned, and the precomputed values used as input data to the network" [Watr90]. Based on this observation and some statistical analysis, it appeared that *five* hidden units (involving 400 links) could have been removed without significant performance degradation - a savings of 38%.

The network got 98% on the training data and 99% (reject level of 1.2) and 98.5% (reject level of 1.5) on the test data. Since recognition was not significantly degraded, as was evident from the relative stability of the recognition accuracy given the varying rejection levels, this suggested a robust network solution. It is interesting that the spectral slope features were largely preserved in the optimization process. This may be due to the category diversification in the spectral slope features, which participated in multiple discriminations and thus, preserved themselves. More importantly, the network solution implied the application of known acoustic cues. The network cued on specific spectral aspects of each sound than attempting to form one large spectral framework for discrimination. This may have occurred because the dimensionality of the space exceeded the minimum necessary requirement for accurate representation of the data set.

Limitations

These networks, although proven to be useful for recognition of specific CV groups, also have some serious limitations, which includes nonoptimal architectures; in particular, the manner architecture. This condition could cause poor generalization resulting in reduced recognition due. Further to this, each architecture was hand tuned to the problem, which is not something that one wishes to have to contend with when developing a recognition system, especially if the data is not readily characteristic of known phonetic groups.

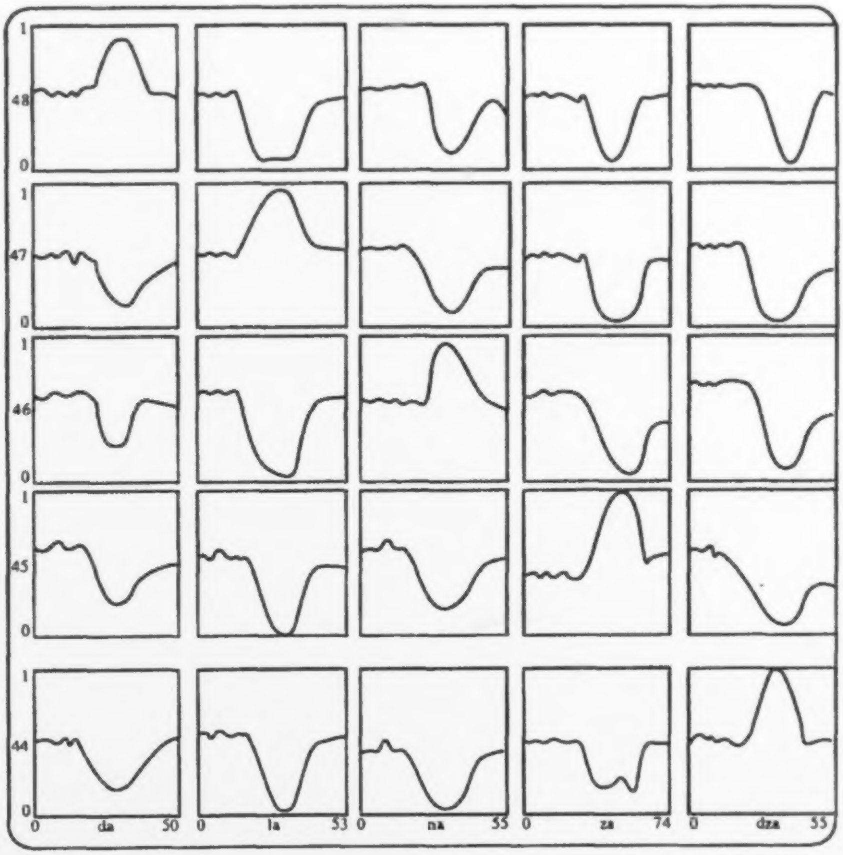

Fig. 5.6' Response of output units (units 44 to 48) of *manner* network optimized on ten training repetitions to an MSE of 0.35% to samples of training data. Display is normalized for segment durations, which are listed as the five ms intervals (after [Watr90]).

In summary, a set of hand tuned TDNN architectures were developed using both the LMS and BFGS algorithms. The networks were invariant to shifts in time. The custom architectures lent themselves to the classification problem by helping in feature representation. On average, recognition was above 95%. Some limitations due to architectural design were present.

5.2.2 Frequency Time Shift Invariant TDNN

Another approach to temporal speech recognition involves introducing a frequency-time context component into the recognition process. In this manner, the network becomes more robust than the standard TDNN shown in Fig. 5.7.

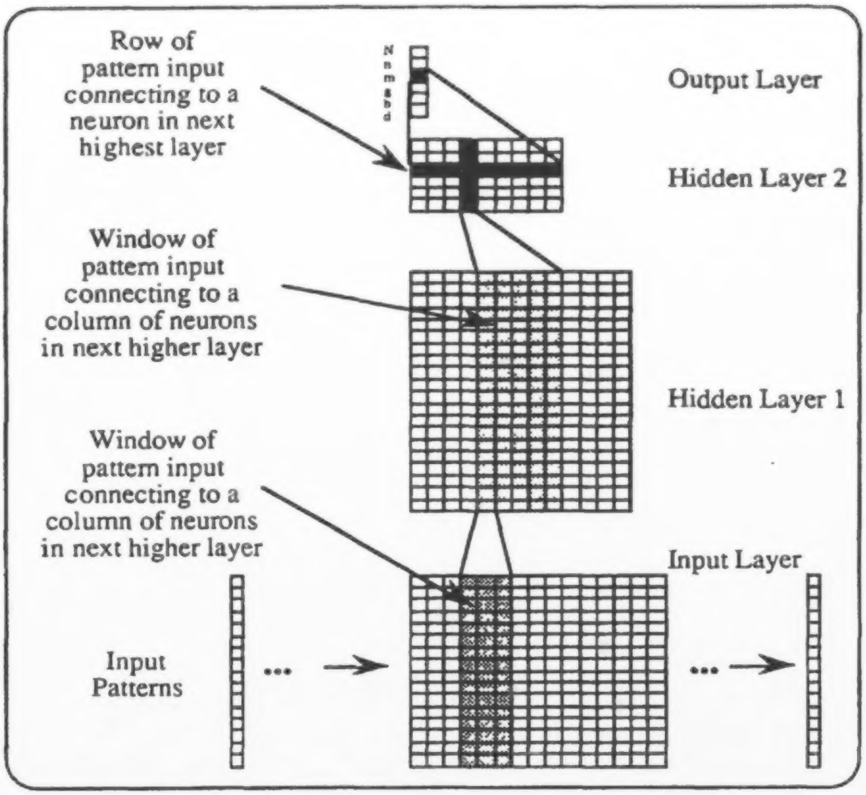
Row of pattern input connecting to a neuron in next highest layer
N
n
m
g
b
d
Output Layer
Hidden Layer 2
Window of pattern input connecting to a column of neurons in next higher layer
Hidden Layer 1
Window of pattern input connecting to a column of neurons in next higher layer
Input Layer
Input Patterns
...
...

Fig. 5.7 TDNN architecture for six phoneme class problem (after [Sawa91]).

This is because it is not only time-shift invariant but also frequency-time-shift invariant. This network is so called a *frequency-time-shift invariant* time delay neural network (FTDNN). These networks are used for continuous time speech recognition. The FTDNN is frequency-time-shift invariant because the weighting structure between the input and frequency based hidden layer is the *same*, which is the same reason that it is time-shift invariant for the regular time based hidden layer. The architecture of this network is given in Fig. 5.8. The left hand hidden layer extracts the time based features and the right hand hidden layer extracts frequency based features. The third hidden layer, which forms higher level abstractions, merges the information coming from the two lower hidden layers. The correct output neuron is obtained by integrating the third hidden layer across a row. This process is both time-shift and frequency-time-shift invariant.

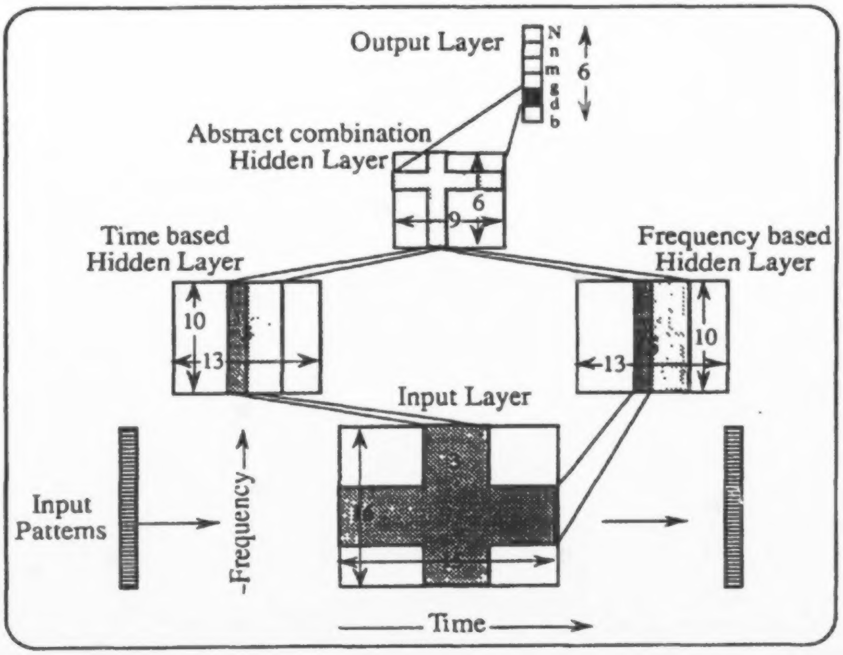

Fig. 5.8 Frequency-time-shift invariant TDNN (FTDNN) architecture for the six phoneme class problem (after [Sawa91])

The original TDNN is *not* frequency-time-shift invariant, which could be more important than being time-shift invariant since frequency-time-shift invariance can reject speech variations in different speakers and their different speaking manners. The increased window width across the frequency axis in the first layer captures the variations in formants of different speakers and absorbs the frequency variations of different speakers. Also, the connections between the input and hidden layer are partially realized by shifting of the input frames. The *parallel* connections have the same weighting values. This makes the network robust to shifts in phoneme positions, i.e., a particular phoneme will not be biased by the different weights established in the parallel connections as it passes through the input layer.

5.2.3 Block Windowed TDNN

The block windowed TDNN (BWTDNN) is a further improvement to the FTDNN. This approach attempts to improve recognition by overlapping both time and frequency variables. As with the FTDNN, the BWTDNN captures local features at the lower layers and more abstract global features in the upper layers. Windowing of the blocks in the lower layers results in an unit of activity in the upper layer. This *focusing* of information gives the invariance necessary in CSR. The BWTDNN operates similar to the FTDNN because the block dimensions also represent time and frequency. The network architecture is shown in Fig. 5.9. Again, the output layer classifies the utterance by integrating the previous outputs of the hidden layer's row. "Especially in continuous speech recognition, respective phoneme recognition rates of 80.8%, and 82.8% for the FTDNN and BWTDNN are attained, which are also significantly better than a rate of 61.8% for the TDNN" [Sawa91].

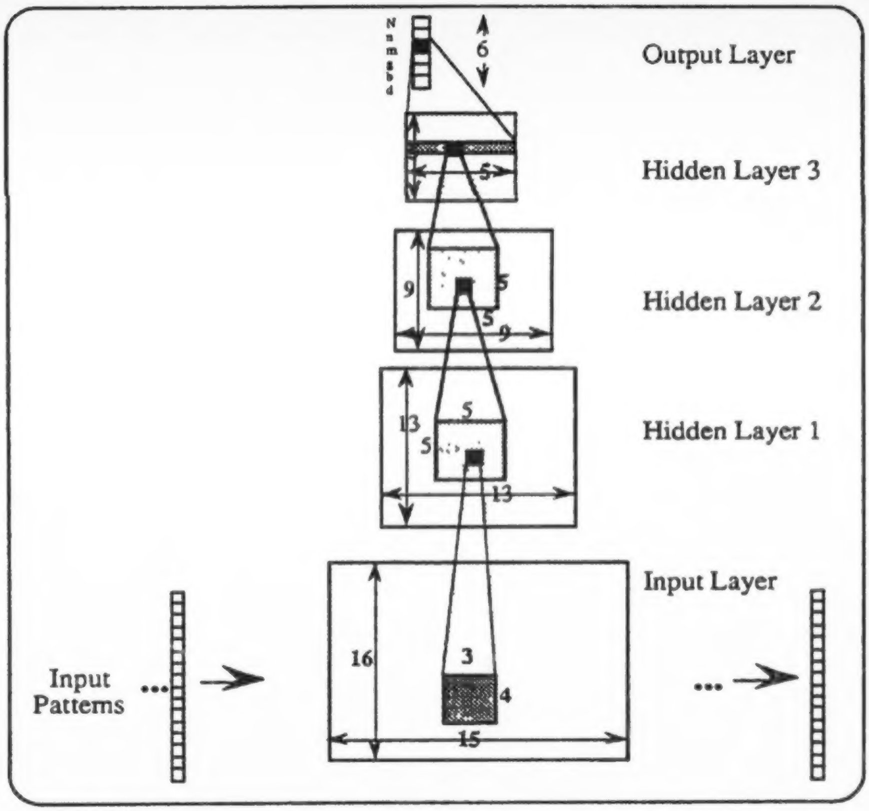

Fig. 5.9 Block windowed TDNN architecture for the six phoneme class problem (after [Sawa91]).

Clearly, both approaches present a significant improvement toward temporal recognition of continuous speech as compared to the standard TDNN. This is because in temporal speech recognition a set of time related patterns must be processed, which these modified TDNNs perform well. Next, these networks are invariant to shifts in time of the pattern input and additionally invariant to frequency-time-shift variations, at least for the FTDNN and BWTDNN. As was mentioned, the ability of being frequency-time-shift invariant may very well be more important than time-shift invariant because of the significant variations between speakers and each individual's unpredictable speech variations. Finally, these networks operate in a continuous fashion. No segmentation of utterances is necessary. When the spectrogram frames are passed through the input layer, the network automatically detects the utterance for which it has been trained, when enough frames are present in the input layer window. With these positive characteristics, better recognition results when compared to the standard TDNN. A common underlying disadvantage is that these architectures, like the TFM's, still need to be optimized to get the best results. If this optimization could be automated then the resulting system may provide better results. This problem is the topic of the next section.

5.2.4 Tempo 2 TDNN

Standard TDNN architectures are not necessarily optimum. A new approach called *Tempo 2* automatically *learns* the optimum architecture. The results of this approach compare well with highly tuned standard TDNNs. By performing this optimization better generalization properties are expected while reducing the amount of memorization compared to nonoptimum networks architectures. Typically, spectrogram frames provide the network input and the frame rate determines the network update frequency. The time delays allow the units of the following layer to capture some temporal context of the activations. The number of time delays decides the amount of monitored context. The amount of required network context is determined, in part, by the following factors:

(i) The amount of temporal context received by a unit.
(ii) The number of weights required to capture the temporal context.
(iii) The delay magnitudes, i.e., how far into the past the network processes patterns.

With more delays, more weights are necessary and thus, more training data are required to train the network. An optimal network solution, which requires a small amount of training data and can capture temporal context, is found by using a feed forward multilayer network with adaptive weights and time delays, and Gaussian shaped input windows with adaptive widths. The time delays and widths (σ - standard deviation) are learned using a similar method to the one used to update the weights in the BP algorithm. The widths (σ) of the Gaussian input windows decide how much temporal context each connection receives. Two Gaussian windows of variable widths and delays, and with different weights are shown in Fig. 5.10.

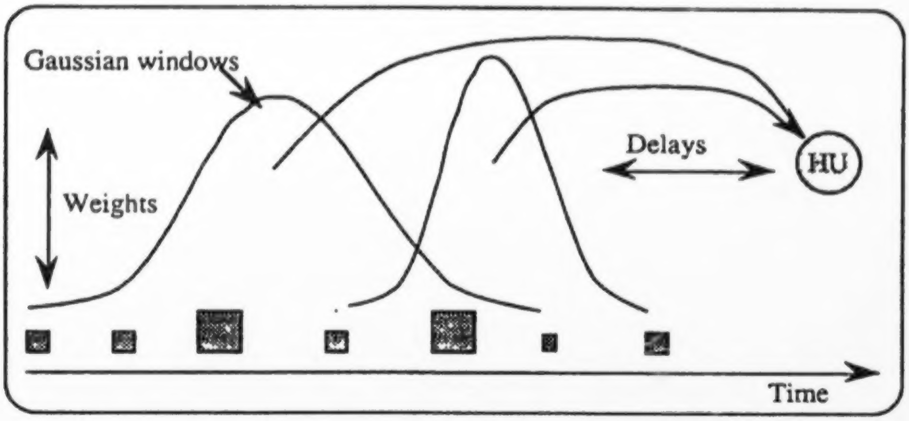

Fig. 5.10 Two Gaussian shaped input windows with different weights, delays, and widths, as used by the Tempo 2 network, are shown to capture more complex temporal dynamics of the input signal, where each box represents a component of the input pattern and the size of the box represents the component's magnitude (after [BoWa91]).

Also, the network grows connections during training, which partially determines the number of weights. The adaptive time delays decide what temporal context is received by each unit (how far into the past each neuron will monitor the input sequences).

Adjusting the Delays and Window Widths

Biologically, a neuron *A* generates a spike (action potential) that has a trailing tail. It takes time *d* to arrive at the next neuron *B*. Neuron *B* will be mostly active at time *t* but only partially active at times $t+1$, $t+2$, ..,$t+N$. The continued activation, which is due to *smearing* of the spike over time, provides local memory to neuron *B*. The Gaussian shape of the input windows (weighting window), with the time delays, artificially simulates a network environment where temporal processing is possible. To work well with speech data the network must be insensitive to temporally misaligned patterns. This is done in the *Tempo 2* network by using a Gaussian shaped input window. The degree of robustness (otherwise described as the amount of temporal context) is directly related to the width of the window. Wider windows provide more invariance to input time shifts but also reduce the time resolution of the unit. An example of how a Gaussian window changes for time delays and window widths is shown in Fig. 5.11.

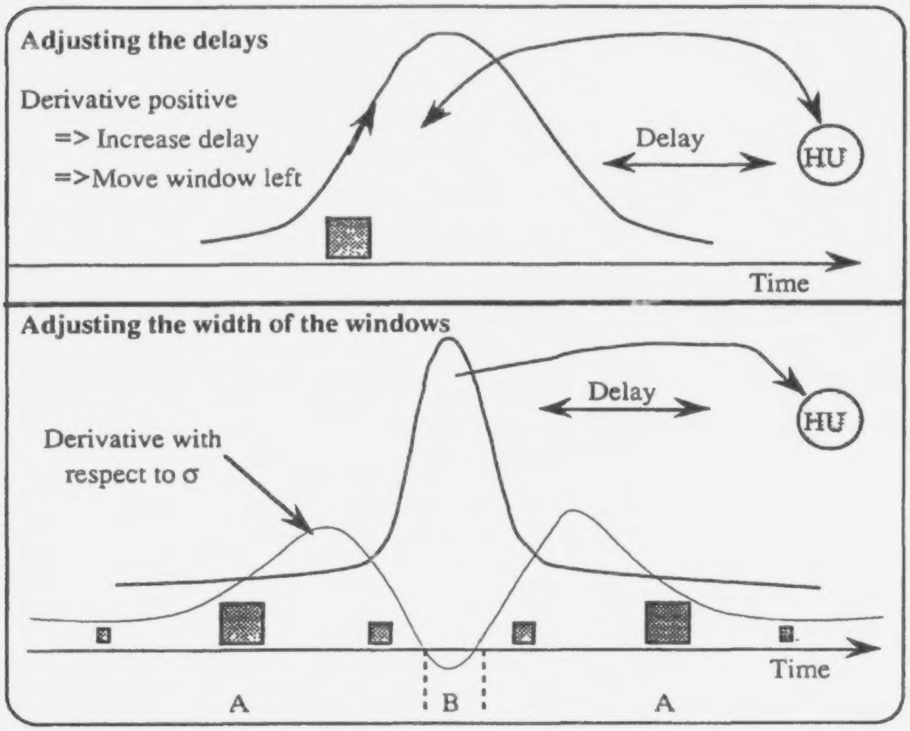

Fig. 5.11 A graphical representation of the learning rules for delays and window widths: The derivative of the Gaussian input window taken with respect to time is used for adjusting the time delays. The derivative with respect to σ (thin line) is used for adjusting the width of the window. More activation in area *A* causes the window to grow while more activation in area *B* causes it to shrink (after [BoWa91]).

Adding New Connections

In optimizing the network architecture, a small network that assuredly is not optimal is proposed, after which connections are added *where they are needed*. Observations at an early phase in training help determine if a connection is necessary and also where it should be included. The changes are restricted to the early training phases because fine tuned adjustments will form as the network settles over time, and if additional connections are introduced, then this would most likely destroy these fine tuned weight adjustments.

Splitting a Connection

When an input window begins to oscillate back and forth across a connection (indicating that time delay magnitudes are changing) that connection is split and the two new connections are retrained. A graphical example of the splitting process is shown in Fig. 5.12. This oscillation "can be interpreted as inconsistent time delays which might be caused by temporal variability of certain features in the speech tokens" [BoWa91].

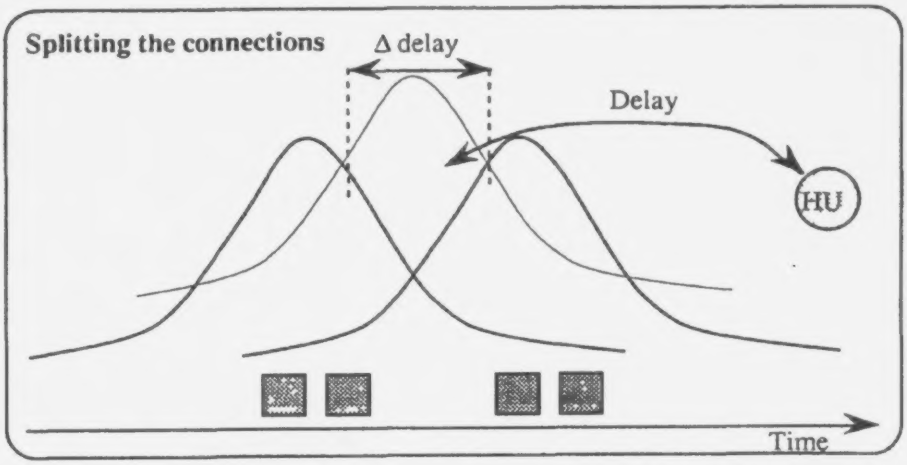

Fig. 5.12 Splitting a connection due to oscillations in the Tempo 2 network: The thin line represents the *old* window and the two heavy lines represent the two *new* windows after splitting. A connection is split into two hew connections if its input window starts oscillating during training (after [BoWa91])

The obvious advantage of the *Tempo 2* network over the standard TDNN is that hand tuning of time delays or architecture (connection strategy and window width) is unnecessary. Also, because of the architectural dynamics that occur during training, the resulting networks are usually one fourth as large as the hand tuned TDNNs. The performance of these networks is comparable to highly tuned TDNNs [BoWa91].

5.2.5 Parsed TDNNs for Varying Phoneme Duration

Parsed, or segmented TDNNs, involve using many hidden layers of various sizes to recognize a set of phonemes. Waibel et al. hope to resolve the scaling problem of the utterance length involved with CSR by including sub-TDNN networks that are tuned to the various phoneme durations in the data set. This network is intended for speaker independent and context independent phoneme recognition. The motivation behind this approach is that "current structures of neural networks must be improved to better cope with the temporal nature of speech. Neural networks usually exhibit poor performance for speech features that are similar, and where the duration information might be the only cue in distinguishing the speech" [WaHa91]. This segregation according to duration has the distinct advantage of dealing more effectively with time varying information. The architecture of the proposed TDNN network is shown in Fig. 5.13.

During training, each TDNN was trained separately and was given a *counter* (reject) category, which would become active if the input did not match the duration of phonemes it was tuned to recognize. This type of network architecture permitted the different TDNNs to *share* responsibility for recognition. In a way, this architectural description allowed specialization since one structure did not have to describe *all* the different classes.

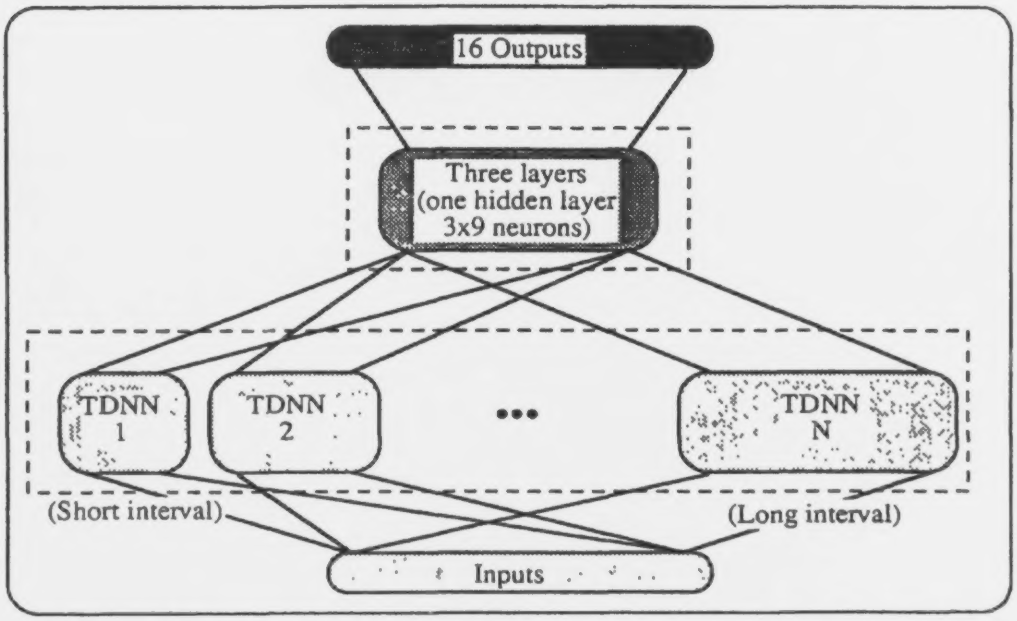

Fig. 5.13 Modified TDNN architecture used for effectively capturing varying phoneme length durations (after [WaHa91]).

It seems reasonable that some irregularities between classes that caused learning problems in a unified network architecture do not exist in this new structure. Therefore, with a reduced amount of training data this network responded like a standard TDNN.

5.2.6 Enhanced Time Invariance using Pattern Prediction in a TDNN

In this approach a recurrent neural network has been modified to include a prediction layer and recognition layer, both of which compose the output layer. The novelty here, lies in the method used to enhance recognition. This is done using a variable known as a *similarity index*. This network can deal with temporal sequences independent of time length. The prediction layer predicts what the next input vector should be on the basis of the present inputs and context. The recognition layer classifies the predicted patterns. The hidden layer contains the information about the next input vector for the prediction and extracted features for recognition. The similarity index is the cosine of the angle made between the actual input vector and the predicted one. The index improves classification by favorably biasing the recognition layer, which is implemented through multiplication of the recognition layer by the similarity index. When two patterns are different, the output of the recognition layer is small, which inturn reduces the possibility of making an incorrect classification. The architecture of this network is provided in Fig. 5.14. This network was trained on the phonemes (/k/, /p/, and /t/). The tests involved recognizing these phonemes using different network architectures.

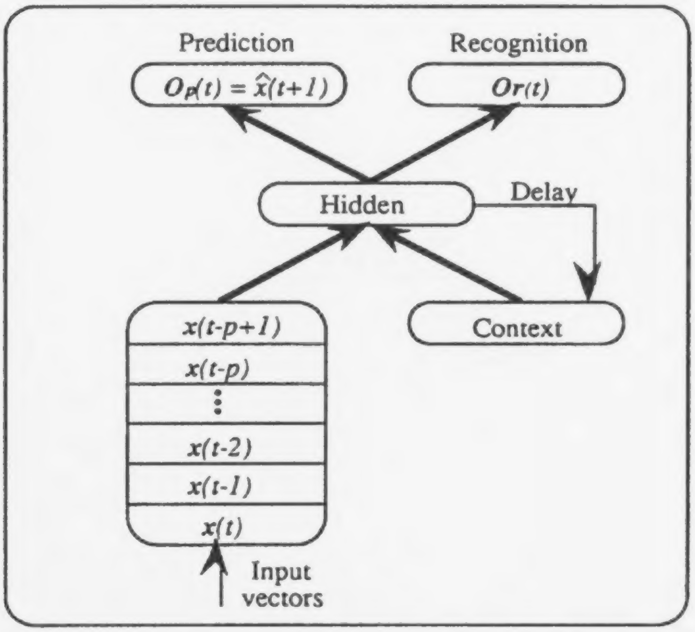

Fig. 5.14 Proposed recurrent network architecture for enhanced pattern prediction. The prediction layer helps support the next correct component in a temporal sequence by predicting the next temporal pattern and assessing its likelihood according to a similarity index (after [FuMa91]).

The results of these tests are shown in Table 5.1. The proportions represent the ratio of correct identifications to attempts. The recurrency of the network is realized through the delay connection from the hidden layer back to the context layer.

Table 5.1 Phoneme Recognition Results for various TDNN Architectures (after [FuMa91]).

Network	/k/	/p/	/t/	Total
Without prediction layer	15/25	24/25	23/25	62/75
With prediction layer	19/25	25/25	24/25	68/75
With prediction layer and similarity index	21/25	25/25	24/25	70/75

The hidden layer contains context information about the next input vector, which it must predict based on context and a window of samples, $x(t)$, ..., $x(t-p+1)$.. The temporal process is realized exclusively through the recurrent connections in the network, which are particularly effective for processing speech signals since these connections supply a delay of sequences providing both dynamic and implicit network memories. Contextual information, which includes temporal changes of sequences, is represented explicitly using memory buffers within an input layer structure. A very interesting approach to encoding the data has been used by [FuMa91]. It involves encoding speech using linear predictive coding coefficients. "A method of feature extraction called linear predictive coding (LPC) is efficient for the representation of temporal sequences" [FuMa91]. This recurrent NN improves recognition by first predicting and then evaluating the next temporal component using a distance metric and cosine operator. The recurrency of the net helps to realize pattern time-shift invariance.

5.2.7 Time State Neural Network

Time state neural networks (TSNN) are another variation of the TDNN that recognize temporal phoneme structures. The performance of these networks exceeded that of the TDNN for continuous speech recognition [Komo91]. With BP many recognition systems using NNs have been proposed but only a handful deal with the temporal structure of phonemes. Some, like dynamic NNs deal with recognition at the word level but their time-shift invariance capability is unknown. The TDNN does time-shift invariance but its architecture "forces it to suppress the temporal structure of the phonemic feature" [Komo91]. The TSNN can be trained using the BP algorithm. The TSNN architecture has four layers and is shown in Fig. 5.15, where every output maps to a phoneme (classifier).

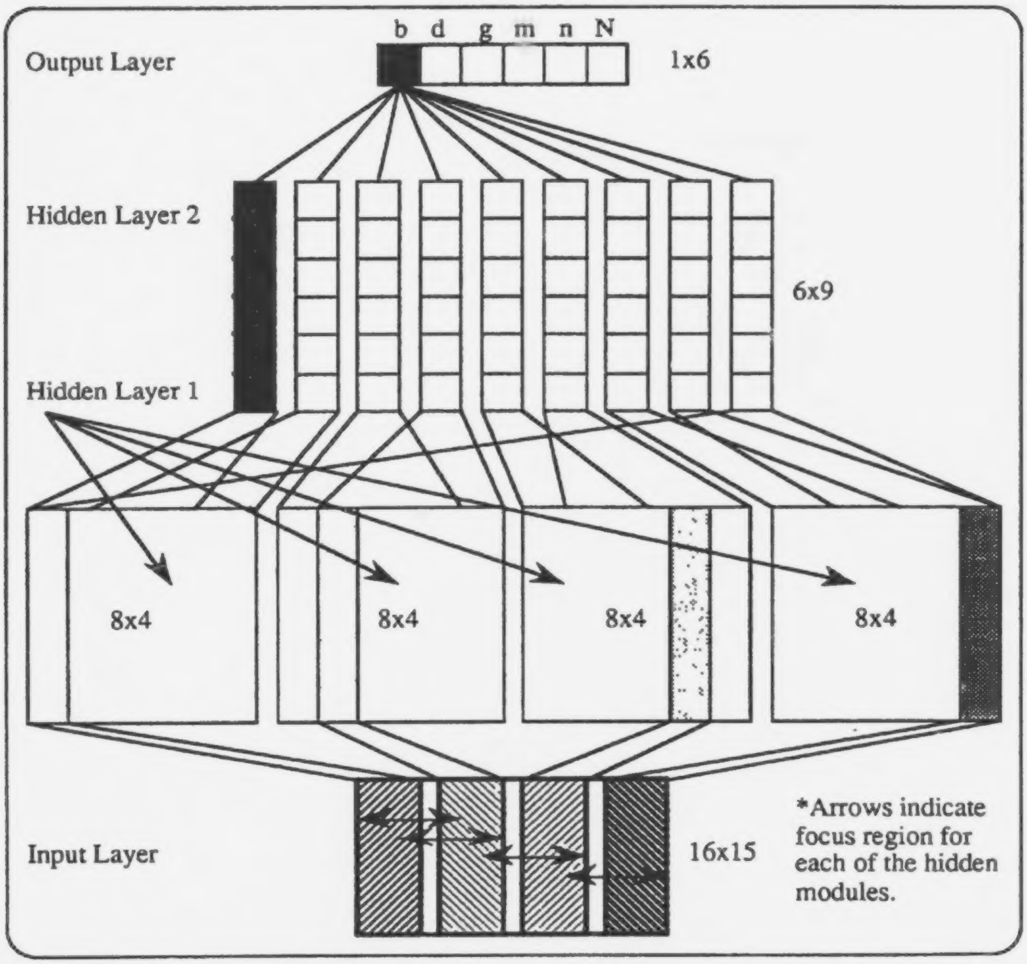

Fig. 5.15 TSNN architecture used to perform the six phoneme classification problem (after [Komo91]).

This network is similar in functionality to the *benchmark problem* of the six phoneme TDNN classifier that is commonly used to compare the worthiness of new temporal TDNN type network architectures. "The first state is considered to capture the transition feature from vowel into consonant, the second to capture the buzz-bar or stable part of the nasal, the third to capture the burst or nasal transition to the next vowel, and the last to capture the next vowel" [Komo91]. The connections between the input layer and the first hidden layer are tie-connected and tie-shifted. Tie-shifted means that the window connectivity overlaps, usually by one column. Tie-connected means that many *n* consecutive columns (window) of the input layer are fully connected to a single column of the hidden layer (this is standard to TDNN connectivity). This network (Fig. 5.15) is not time-shift invariant because it is only tie-shifted and tie-connected between the input and first hidden layer.

The next architecture, shown in Fig. 5.16, is tie-shifted and tie-connected between every layer. The connectivity is very unusual. The input layer is split into thirds. The left third of columns are connected to the left hidden layer block and the middle third of the input layer is connected to the middle hidden layer block, etc.. This splitting of the domain space helps to represent discontinuities and complex regions better than a single fully interconnected hidden layer. As will be seen however, it has a disadvantage of having a worse representation of shorter utterances due to the divisions, i.e., due to scaling and other related issues, the representation is degraded across the hidden layer subnetworks. Longer utterances (words and phrases) are better represented with this form of architecture, as well as being insensitive to scaling. The architecture connectivity is again done by using multiple consecutive columns (windows) of the input layer that are fully connected to a single column of the hidden layer. These windows overlap usually in an adjacent input layer column as seen by the domain arrows in Fig. 5.15. This type of connectivity is replicated between consecutive hidden layers, except that the widths of the windows may vary to establish different temporal relationships. In a sense, different segments of the input are *focused* to a modular hidden layer. An input frame, typically spectral coefficients, is still shifted through the entire input layer, but the difference is *who* reads the input, *where* it goes, and *how* it is connected.

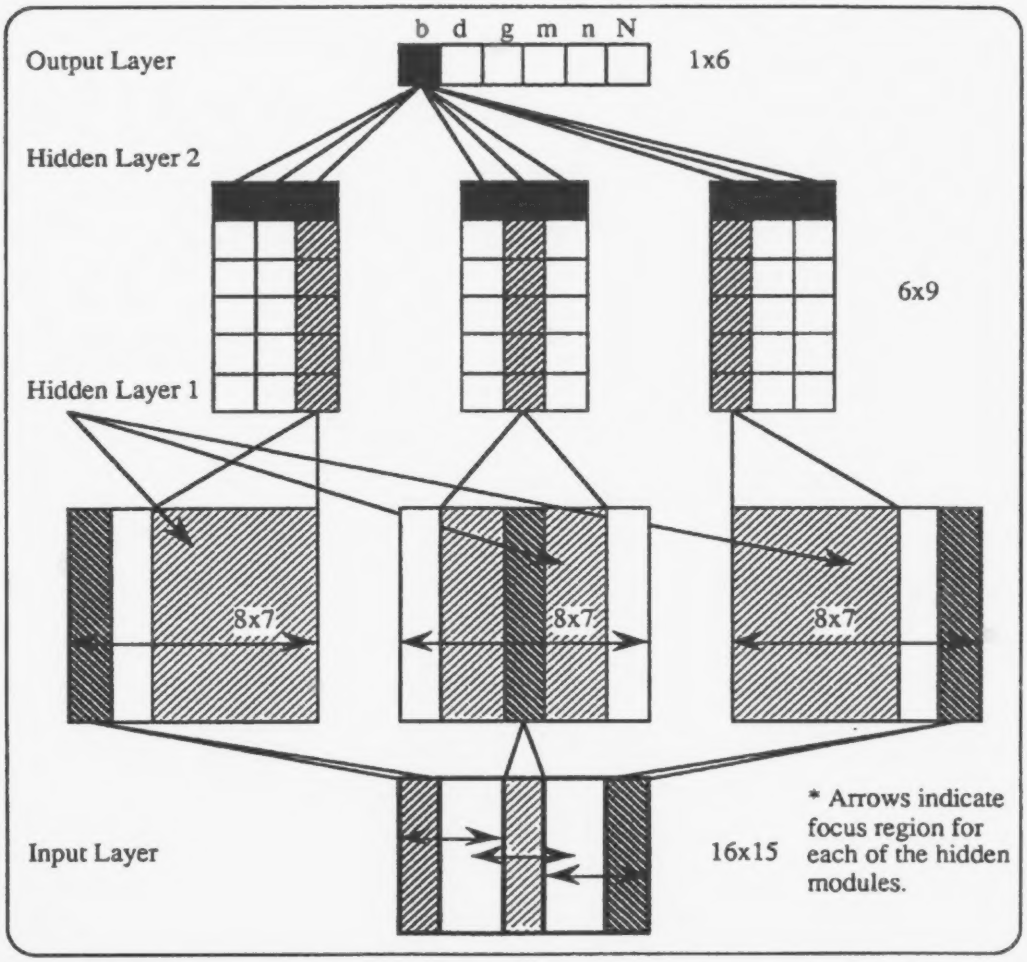

Fig. 5.16 An improved TSNN architecture by using modularity in the hidden layer. This architecture is time-shift invariant due to the common weighting structure between the input and hidden layer. It focuses its attention on the various phoneme transitions using the modular architectural design and connectivity (after [Komo91]).

These architectures (Figs. 5.14, 5.15, 5.16) were tested using the ATR speech database. Isolated utterances, continuous utterances including short phrases, long phrases, and sentences were used. The benchmark comparison was against the standard TDNN architecture designed for the six phoneme classification problem. The results of these experiments are shown in Table 5.2. Although the TSNNs are not proficient at recognizing isolated utterances, they excel at the continuous problem class. The reasons for the unusual behaviour, i.e., poor performance on isolated speech and good performance on continuous speech is not covered, but the results suggest that these architectures help doing temporal and continuous recognition through, in part, their connection strategies.

Table 5.2: Phoneme Identification Performance Using Time State Neural Networks (after [Komo91]).

*Bold indicates the position of training data

Input Data Style		150 ms fixed				Normalized by Duration	
Utterances	Neural Networks	TDNN	Simple TSNN	Simple TSNN shift data training	TSNN with tied-connections in the hidden layer	TDNN	Simple TSNN
Isolated Word Utterances	-20 ms	91.2	52.7	81.6	58.7		
	-10 ms	94.7	86.7	**95.7**	92.6		
	0 ms	**95.7**	**97.6**	**97.0**	**97.0**	**95.6**	**98.0**
	+10 ms	94.1	84.4	**95.0**	89.9		
	+20 ms	85.9	56.5	77.5	65.8		
Continuous Utterances	Short Phrase	76.6	82.7	79.9	79.0	74.6	83.8
	Long Phrase	75.9	77.6	77.6	77.7	74.8	81.6
	Sentence	61.8	70.7	71.1	69.0	58.5	72.0

5.3 Temporal Backpropagation Neural Network

A specialized BP neural network is presented that uses context in its architecture to perform temporal pattern recognition. The key feature in this architecture, like that found in many networks of this nature, "is a layer of self-connected hidden units *that integrate their current value with the new input at each time step to construct a static representation of the input sequence*" [Moze88].

5.3.1 Focused BP

Most static connectionist networks are not good at dealing with temporal patterns because they do not have context. A specialized version of BP has been developed that recognizes temporal sequences. The architecture avoids two deficiencies contained in other standard BP sequence recognizers; first, it reduces the problem of temporal credit assignment by *focusing* the back propagated error signal, and second, it eliminates the need for a buffer to maintain all the past temporal information. This is because it generates incremental activity *traces* that hold the necessary information used in the weight updates. The hidden architecture is based on an old form of temporal neurons called wickelphones [RuMc86]. These units provide a very good representation of the temporal information contained within the network [Moze88].

Architectural Motivation

Backpropagation, in its standard format is not suited to deal with the problem of temporal recognition because its architecture cannot perform temporal integration. A sequence of vectors presented sequentially in time is required to stimulate a desired response as shown in Fig. 5.17. A popular approach to deal with this problem was to create a buffer that held the *n* most recent elements of the input sequence. This approach was implemented using shift registers or delay lines. This strategy converted the temporal problem into a spatial one, which made using the connectionist architectures suitable since they are good at solving spatial recognition problems.

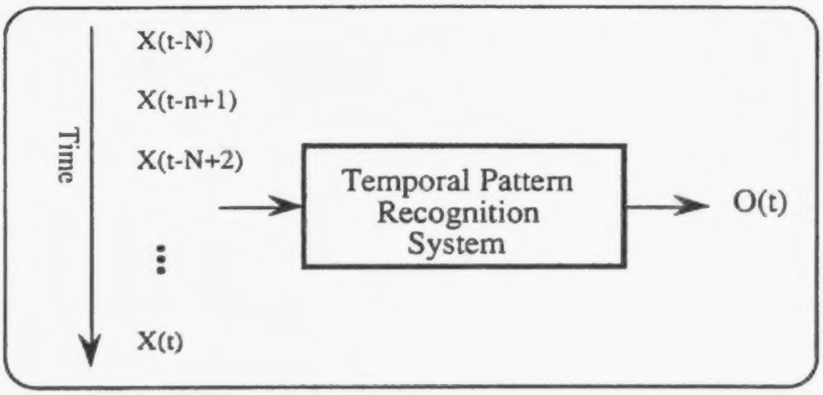

Fig. 5.17 Abstract characterization of the temporal recognition task. $X(t)$ is the input pattern at time t and $O(t)$ represents the output pattern (after [Moze88]).

The buffer approach had four serious drawbacks though. The first was that the buffer had to be long enough to fit the longest sequence. This may not have been an immediate limitation, but one that still had to be considered. The second problem involved the number of buffers that were used to store the pattern history, which made the data available simultaneously. Additionally, the recognition task became more computationally intense since a serial machine was used. Third, since all the buffers were necessary for processing, many network connections had to be established. As the number of buffers grew, so too did the number of connections, but faster. This also meant that as more weights were added, more training examples had to be included to guarantee statistically reliable network weights. Finally, the patterns were learnt in a static fashion, so, any translation in time caused significantly reduced recognition results. It necessarily meant that many translations of the input also had to be trained, which resulted in increased overhead and a greater load on the network weights. Because of these deficiencies, methods were devised "to avoid turning intrinsically temporal information into spatial information" [Moze88].

Although the buffer model had these problems it did have two essential components that made it useful. First, it could process past information and second, it could combine current context (memory) with current input to form subsequent context. This relationship is shown in Eq. 5.3 where *C(•)* is context and *X(•)* is an input pattern. The resulting architecture shown in Fig. 5.18 provided the insight to a solution.

$$C(t+1) = f(C(t), X(t)) \tag{5.3}$$

The architecture shown in Fig. 5.18 describes a set of context fixed weights that are connected to themselves, which is the proposed solution. This structure with input, *X*, and context, *C*, is expressed by Eq. 5.4.

$$f(C, X) = k_1C + k_2X, \text{ (where k1 and k2 are constants)} \tag{5.4}$$

This network's independence from spatial time representations, buffers, and temporal sequence length make it a very good temporal recognition network. Still the mapping is fixed. An optimum solution results if this rigidity was removed; the solution to this is discussed shortly. Backpropagation, which is a feedforward network, must be altered to implement the recurrent structures necessary in contextual processing. The temporal context was implemented by way of recurrent links. Both Figs. 5.18 (a) and (b) are really the same except that Fig. 5.18 (b) has been *unfolded in time*.

Training a network of this type has two drawbacks. The first is that the architecture must be replicated in time and the second is that unfolding results in deeply layered architectures. These networks are prone to widely dispersed error signals, due to the increased depths, which results in longer training periods because the neurons at the lower layers will not get the correct amount of credit/blame from the error signal.

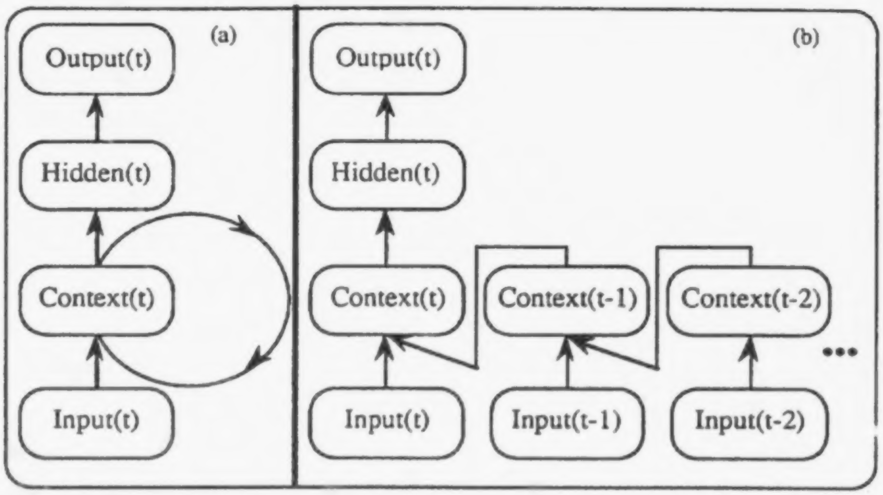

Fig. 5.18 Unfolded BP architecture. (a) Desired architecture of a contextual system with temporal memory. (b) Unfolded architecture of (a) can be implemented using buffers (after [Moze88]).

The architecture shown in Fig. 5.18 (b) must exhibit the following properties:

(i) It must statically encode the input pattern and temporal relationships between patterns.
(ii) It must encode varying length sequences.
(iii) It must provide a natural basis of generalization.

The *Wickelphones* [Moze88] encode each sequence element according to its local context within the sequence. Wickelgren (1969) proposed a context sensitive scheme where each wickelphone responded to a particular phoneme arrangement. For example, "if the word *explain* had the phonetic spelling /eksplAn/, it would be composed of the wickelphones $_{\_}e_{k}$, $_{e}k_{s}$, $_{k}s_{p}$, $_{s}p_{l}$, $_{p}l_{a}$, $_{l}A_{n}$, and, $_{A}n_{\_}$ (where the dash represents a word boundary)" [Moze88]. The word is activated according to the distributed pattern of these wickelphones over the input. Unfortunately, there are too many wickelphones; even if the non-English combinations such as /pkt/ are not included, the number is intractable.

To get a pseudo wickelphone environment in the BP network that satisfies the four conditions while *not* using the unfolding scheme, an internal context layer representation was devised. The first step is to present multiple time frames of the input, i.e., patterns at times *t-N, .. ,t-1*, and *t*. During these presentations, the context elements detect and represent in their weights, the temporal relationships between consecutive input frames. Once the context units have triggered on something in the input sequence they remain active. The context units employ *recurrency to themselves* using the relationship given in Eq. 5.5 where $c(\bullet)$ represents the activity of a context unit, d_i represents the decay weight that is used to remove spurious noise in the temporal input sequence, $s(\bullet)$ is the standard sigmoid activation function, and $net_i(t)$ is the net input to a unit.

Early in training, the sigmoid for the context node is shifted since this provides richer dynamics. Learning increases because both domain limits are non-zero, i.e., {- 0.5, 0.5}. Later, the sigmoid is switched back to its typical range of {0, 1}. A useful relationship, producing the *activity traces* (Eq. 5.8), is derived between the standard error signal (Eq. 5.6) and decay factor relationship (Eq. 5.7). The traces, which update the network in the forward pass, have the same effect as the BP updating step. The algorithm details are in [Moze88]. A description of these dynamics is shown in Fig. 5.19.

$$c_i(t+1) = d_i\, c_i(t) + s[net_i(t)], \text{ where } net_i(t) \equiv \sum_j w_{ji}\, x_j(t) \tag{5.5}$$

$$\delta_i(t) \equiv \frac{\partial E}{\partial C_i(t)} \tag{5.6}$$

$$d_i = \frac{\partial C_i(t)}{\partial C_i(t-1)} \tag{5.7}$$

$$\delta_i(t-1) \equiv \frac{\partial E}{\partial C_i(t-1)} = \frac{\partial C_i(t)}{\partial C_i(t-1)} \bullet \frac{\partial E}{\partial C_i(t)} = d_i \delta_i(t) \tag{5.8}$$

This recurrent relationship used in the BP network is what Mozer calls a *focused* network because "whatever error is propagated back to the context unit at time *t stays* within that unit as the error is passed further back in time, in contrast to a full recurrent network where the error is redistributed among the context units with each backward pass due to cross connections between units" [Moze88]. No dispersion of the error signal occurs here as does in the recurrent architectures. Connectivity in the context layer is limited to one-to-one recurrent connections and is linearly integrated over time.

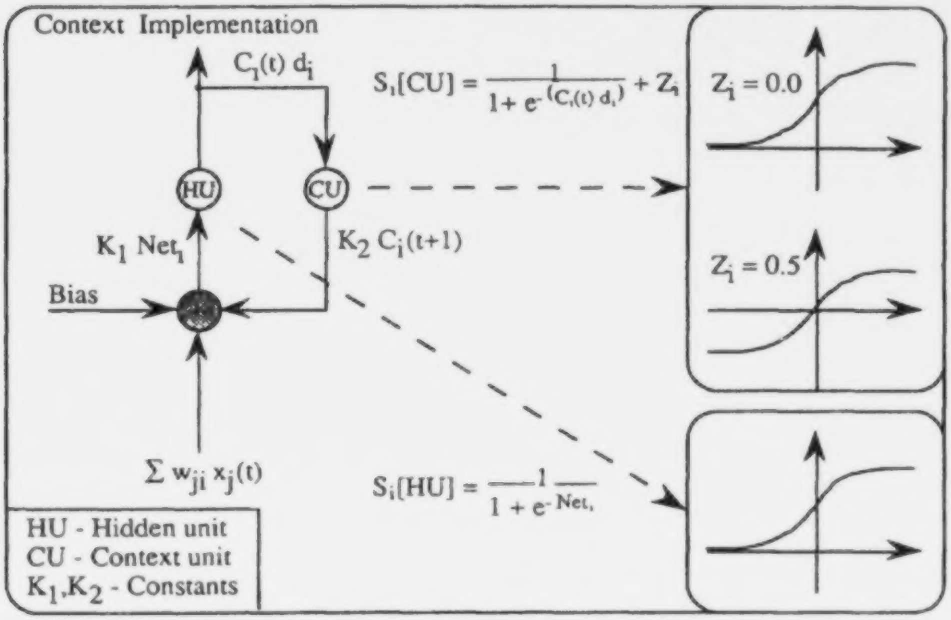

Fig. 5.19 Activity trace and context implementation used in focused BP. The activity traces eliminate the need for replicated input buffers used as context. They are used to update the network in the forward pass.

Experiments and Results

The network was tested against the fully recurrent NN on regular verbs such as *depend* /dEpend/, *guide* /gId/, *include* /inklUd/, etc.. The results showed that both networks learned this task easily. Also, the delay units were all near zero, which suggested that no contextual history was necessary. To increase the difficulty, the verbs were reversed. For example, the word *explain* /eksplAn/ now is presented as /nAlpske/. The relevant information comes at the beginning of the sequence and must be retained until the completion to decide the correct identity. The result of this test, averaged over fifteen trials and using random initial weights for each trial, is shown in Fig. 5.20.

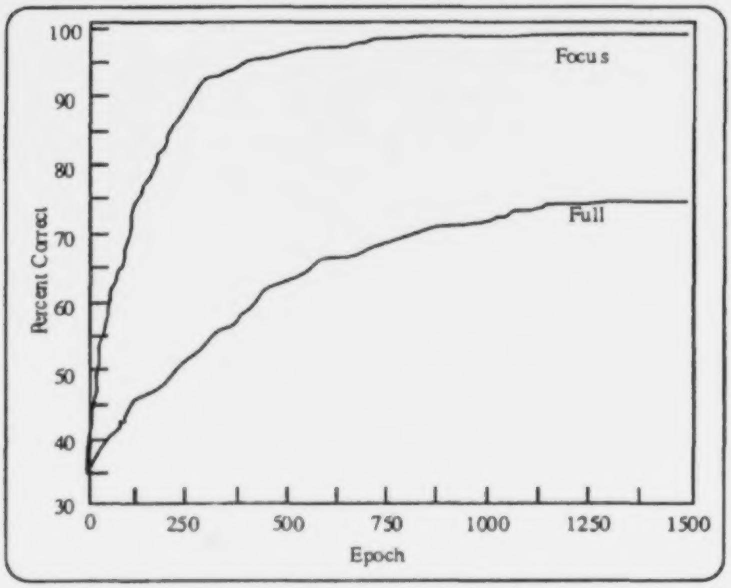

Fig. 5.20 Mean performance on the reversed regular verb task as a function of learning epoch for the focused and fully recurrent architectures (after [Moze88]).

The best single run performance of the full recurrent network only reached 75%, with 98% for the focused. The fully recurrent network could not learn this task. This result suggests that the backpropagated error signal becomes too dispersed in the fully recurrent network for high level abstractions to develop. Recognition on the novel cases was about 54% for the full recurrent network and 96% for the focused network. Both architectures found it much harder to learn temporal relations as the pattern sequence length increased, i.e., when they had to remember more before making a decision. The effect of the focused algorithm on this difficult problem resulted in increased training while the fully recurrent network simply could not learn the problem. The focused architecture is also superior in two other respects; it does not require each unit to maintain a stack of intermediate activity levels and it is less computationally intense because it does not have to back propagate the error signal (focused architecture requires about two thirds as many floating point operations as a similar size standard BP network) if activity traces are used [Moze88].

The next subject to discuss is how well the network scales. More depth is required to scale the input of a fully recurrent network; the consequences of which, have been described. With the focused architecture though, instead of increasing the depth to scale the input sequence, increase the breadth. Since this does not dilute the focus effect of the error signals the network generalizes better and trains in a shorter period as compared to the fully recurrent network. Thus, although the sequence length is increased, generality within the network structure is not sacrificed.

One problem that Mozer pointed out is that he did not know how the network would respond to longer length sequences in *other* respects. He said, " . . . it may be appropriate for somewhat short sequences, it is unclear how well the approach will work on long sequences in which very little information is contained in a single sequence element, for example, a speech recognition task with the time domain waveform as input" [Moze88]. Another criticism of the network concerns the use of input buffers to train context. The input buffer is not strictly necessary. Still, in a network without buffers, a unit cannot tune itself sharply to input ***A*** followed by input ***B***, but not to either ***A*** or ***B*** in isolation. The focused architecture buffer window specifies a temporal window over which nonlinear interactions occur. The last problem with the focused architecture is the delay weight values. If these increase in magnitude beyond one an instable condition occurs. This can be limited (less than one) through dynamic normalization, but results in longer training periods. This normalization procedure also results in stable systems and better generalizers.

5.3.2 Adaptive Temporal BP

In this generalization of the feedforward BP network, all connections have adaptive time delays and weights. This permits smaller and more efficient mappings. Both are trained using gradient descent using either *epoch* updating or *on-line* training. The application of this network was to predict the next six samples of a chaotic signal derived from the Mackey-Glass differential equation. The networks are static, responding immediately to a single input with the appropriate output. Also, they are fed continuous time varying input. For problems like speech recognition that are temporal in nature and involve patterns too long to be recognized immediately pose unsolvable obstacles to these networks. Spatio-temporal recognition is also possible. One way to recognize these temporal patterns is to delay the input patterns as shown in Fig. 5.21. This delayed input provides history to the static network.

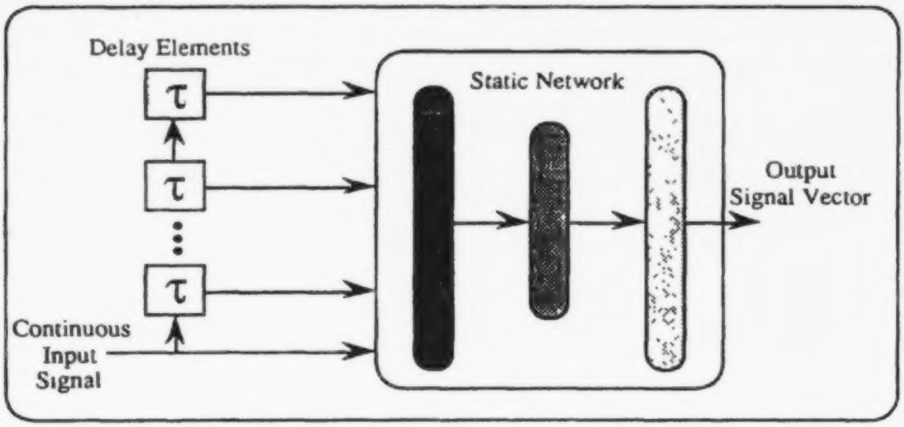

Fig. 5.21 Spatialized temporal recognition. The input signal is fed into the static window with the *N* past input frames. There are no time delays within the network (after [DaDa91]).

"For networks to respond to these temporal patterns, it is necessary to have *additional* delay elements located within each layer . . ." [DaDa91]. This is a well established and biologically plausible approach since time delays are present in axons and synapses of biological neurons.

The Temporal BP Network

By *unfolding* a temporal structure spatially the problem of temporal context representation is solved. Additionally, including adaptive weights and delays helps develop the necessary temporal representations within the weight structures. If both input and output vary continuously in training and recognition operation, then segmentation can be neglected initially. Ideally this is desired, but somewhere along the way part of the system, whether it is at the preprocessing stage or implicitly managed within the recognizer, must deal with the problems of segmentation and coarticulation.

Training the network involves minimizing the difference between the output trajectory and the target trajectory. Only nonrecurrent networks are used; there is no feedback from an output of a node to its input. Still, connections *across* multiple layers and multiple connections *between* nodes are permitted. The output and input may either be a simple pattern or a sequence of temporal patterns. This type of architecture is shown in Fig. 5.22.

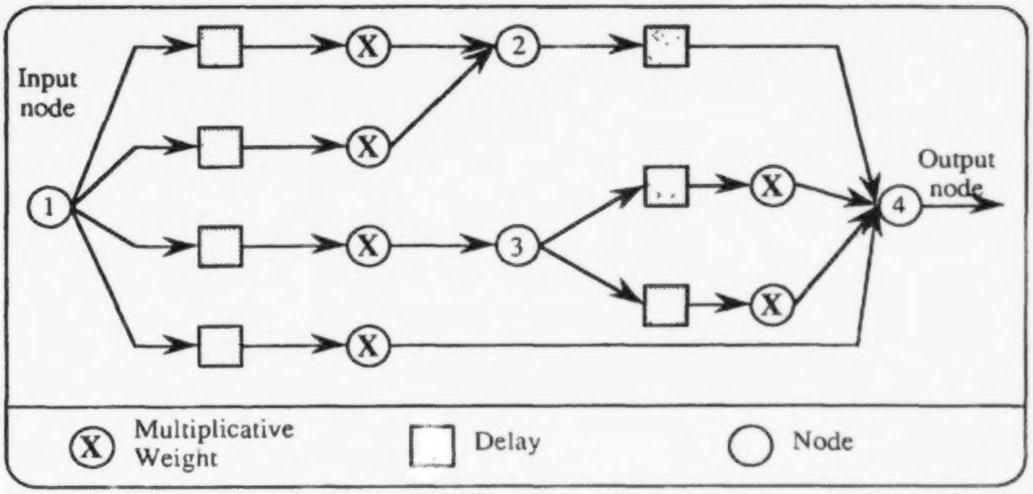

Fig. 5.22. Architecture of a modified temporal BP network. All connections have adaptable weights and time delays. No recurrent connections are used (after [DaDa91]).

If temporal patterns are chosen then the input signal must be long enough that when it finally propagates through the network the correct output pattern is present. It is all matter of timing really. In terms of learning, you may use either the standard epoch rules or *on-line* learning. For on-line learning, a good *approximation* to the exact learning rule is realized,which is significantly less computationally intense if the learning rates are kept small for the weights and time delays. A problem with on-line learning is that "at high learning rates, the time delays can cause the adaptable parameters to overshoot their optimum values, lending to instability" [DaDa91]. On-line learning can detect this and reduce the possibility of this occurring. On-line learning is also advantageous over epoch learning because the weight changes, Δw_{ijk}, accumulated over an epoch can become large, especially if the training signals are long. The large weight changes exist because the gradient is not followed optimally due to quantization of the updates. This can be avoided by reducing the learning rates, which effectively scales down the Δw_{ijk}, and therefore, lengthens training. The last advantage of *on-line* learning over regular epochal learning is that it can adapt to *new* (additional) training data *dynamically* by continuously following the gradient.

The Mackey-Glass time series prediction problem has become a popular benchmark for temporal supervised learning [DaDa91]. Networks trained with momentum exhibited lower prediction errors, though the on-line prediction errors were higher. Day and Davenport speculate that momentum may be detrimental due to short-lived anomalies in the data set, but result in better approximations after training has been completed. Networks trained with momentum produced much more accurate spectra than networks trained without momentum. The main difficulty in implementing a TBPNN is the coordination of the time delayed error gradients and feed forward signals. The proposed solution to this is to assign a fixed *age* to each node, and add delay buffers as necessary to ensure that all signals required for learning are of the proper age. In summary, a TBPNN recurrent type network was developed using adaptable time delays and weights that illustrated the ability of this network to learn complex temporal behaviors. The use of momentum produced better results.

5.3.3 Forced Simple Recurrent BP

This model is designed to recognize a series of symbols comprising a known sequence such as a phoneme or word. The network uses the backpropagation algorithm. The temporal integration, necessary for sequence recognition arises from the context developed in the hidden layer, which is fed back one time step later with the current input symbol. This network evolved from the simple recurrent network (SRN) but had to be improved due to a serious problem arising from a particular situation involving the failure of the hidden layer to represent uniquely the context and, therefore, predict the next symbol. The result of this failure became a local minimum. The following discussion describes the forced simple recurrent network (FSRN) and describes the limitation of the SRN and then why and how the FSRN solved this problem. After the discussion, a few experiments exposing the limitations of the SRN are presented and contrasted to the FSRN's ability to successfully solves the context representation and prediction problem.

In these networks, the temporal patterns are represented as a set of related symbols. "Patterns having a temporal dimension can often be represented as sequences of known patterns, or symbols. The processing of temporal patterns can then be characterized as either sequence recognition or sequence generation" [MaNo92]. In a recurrent NN, the network takes an input symbol and its context within the sequence, which is generated from all predecessors symbols of that sequence, to generate the output symbol. Now in a sequence generation problem since the next symbol is known the recurrent NN can easily be trained. In this way, "the temporal pattern recognition problem is converted to a problem of pattern generation" [MaNo92]. A network such as this can be trained to predict the next symbol in a sequence using only a subset of the symbols. To produce the prediction though, the SRN creates an internal representation of the components it has seen so far. These symbols are called the *context* with respect to the current symbol. Sometimes, the SRN cannot form this context representation so an improvement to the meaning of *context* was developed, which is called the FSRN. The standard SRN is shown in Fig. 5.23 (a) and the FSRN is shown in Fig. 5.23 (b). The SRN input units are forced to represent only the successive symbols of the input sequence.

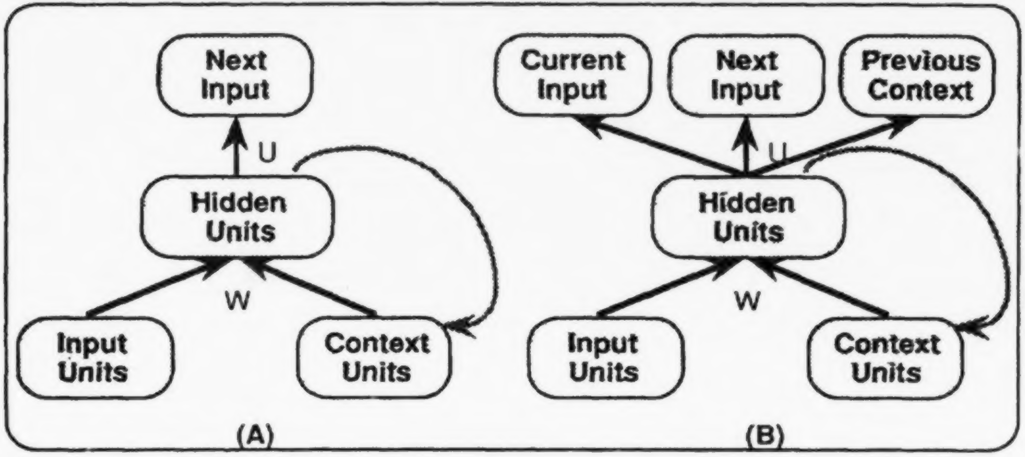

Fig. 5.23 The SRN and FSRN. (a) The standard simple recurrent network does not have enough information to solve the problem involving the undesired merger (local minimum) of two context branches. (b) The forced simple recurrent network overcomes this limitation by creating a more robust context representation that forces the context paths to remain separate and unique to their patterns (after [MaNo92]).

After each interval the hidden layer activation is copied into the context units, which are then fed back into the input layer with the next symbol. Instead of being a pure classifier, the output of the SRN provides a distribution of output values from which a prediction symbol is chosen. At first, the context values have no past input to provide to the input layer with the first symbol so they are set to a special state known as q_o. The

SRN trains using the BP algorithm. Not only does the output act as a classifier but the input also does since given N input units, a symbol input pattern will have only a single value high while the other symbols are low. The context is developed when "a particular sub-sequence, or *context*, is always followed by a particular x "$\in S$, where S is the series [MaNo92]. Here, the network learns to associate a high value with x and low probabilities with all other symbols and where a common context is followed by different symbols, those different symbols that follow the common context will have higher (although not maximum) probabilities than those symbols that do not follow the common context. A sequence is *accepted* if all the symbols are correctly predicted and it is *rejected* if *any* symbol is not correctly predicted.

Context Representation Process

The process of encoding the context of the prior sequence to the current symbol involves many steps. Initially, the activation pattern of the hidden layer is determined and stored (as a state) for every symbol. Since this includes much redundancy these states are compared and, if they are close, then two states are combined. The comparison is done on a unit to unit basis and the threshold is such that every coincident pair must not differ by more than five percent for a match to succeed.

SRN Limitations

Early in training each symbol begins to form a representation in the hidden layer. Since this output at time t is used with the next symbol at $t+1$ the encoded context gradually begins to form a mapping to the previous symbol. The next level of encoding is intended to form mappings of symbol combination, until finally, the entire sequence is mapped into the hidden layer. The problem or deficiency arises in a special case where the mapping fails.

During training, if a particular aspect at time t is not encoded in the hidden layer then it becomes lost since it will not be propagated at times greater than t through the context process. This essentially breaks a vital link in the chain when it comes to recognition, which inevitably may lead to recognition failure. Lets suppose for a moment that there are two different states, q_i and q_j and further suppose that each have the same output but different contexts leading up to these states. If this is the case, the hidden layer may very well succeed in using the same representation for these two different contexts. If this happens the SRN will never develop different hidden layer encodings for the different contexts and will remain in this local minimum. If the SRN is to recognize these two sequences as different, and yet provide identical output predictions, it is going to need another dimension of the data to distinguish the sequences apart from one another.

A first guess may be to try to enhance the differences between these two contexts using momentum. "The use of momentum could amplify small differences in context and take the network out of the local minima. However, the use of momentum is not generally effective in taking the network out of a local minimum" [MaNo92]. The solution to this problem resulted in not going with the momentum theory since it did not appear to help much anyway. Instead, the number of hidden and output units were increased, which provided more dimensionality to the space and increased the robustness of the context representation. The context robustness involves "requiring the output units to show the previous context and current input" [MaNo92], besides the regular prediction (refer to Fig. 5.23 (b)). In the FSRN, the representation process initially began the same as the SRN with all states initially being separately represented in the hidden layer. But the difference arose when the context path merging problem appeared. This is because, in the FSRN, a unique context is maintained due to the hidden layer being *forced* to learn the current symbol too. This is where the uniqueness is created and, therefore, where the problem is solved since the current symbols are different though the symbol prediction is identical. To expose the problem more clearly a few experiments conducted by [MaNo92] are included.

Experiments

These experiments show the problem of the SRN and its solution using the FSRN. Both networks used similar network topologies, identical initial starting weights and were also trained for an identical number of epochs. The random order of presentation was also duplicated for each network. The first experiment involved training the SRN and FSRN to recognize a seven state system as shown in Fig. 5.24 (a). The beginning and end state are labelled as *S* and *E*, respectively. The letters *X*, *Y*, and *P* are intermediate states. The networks were trained to recognize the following sequences:

SXPE, SXPXPE, SXPXPXPE
SYPE, SYPYPE, SYPYPYPE.

Both the SRN and FSRN were given three hidden units. After training only the FSRN was successful at recognizing the sequences. After examining the SRN's hidden layer it was determined that the symbols X and Y had been assigned the same representation, as described by the SRN limitation. The development leading up to this result is shown in Fig. 5.24 (b-e). In this figure, it is clearly shown that the two separate context developments eventually were given the same hidden layer mapping because, although the original contexts were different, at one point the next prediction, *P*, was identical, leading to state seven. This commonality led first to a representation of *X* and *Y* being the same and later to their identical representation in the hidden layer. This experiment clearly shows the need for some *forced* information to separate the different contexts in the space. In the second experiment longer sequences were used. The architecture used here is shown in Fig. 5.25. Four hidden units were used.

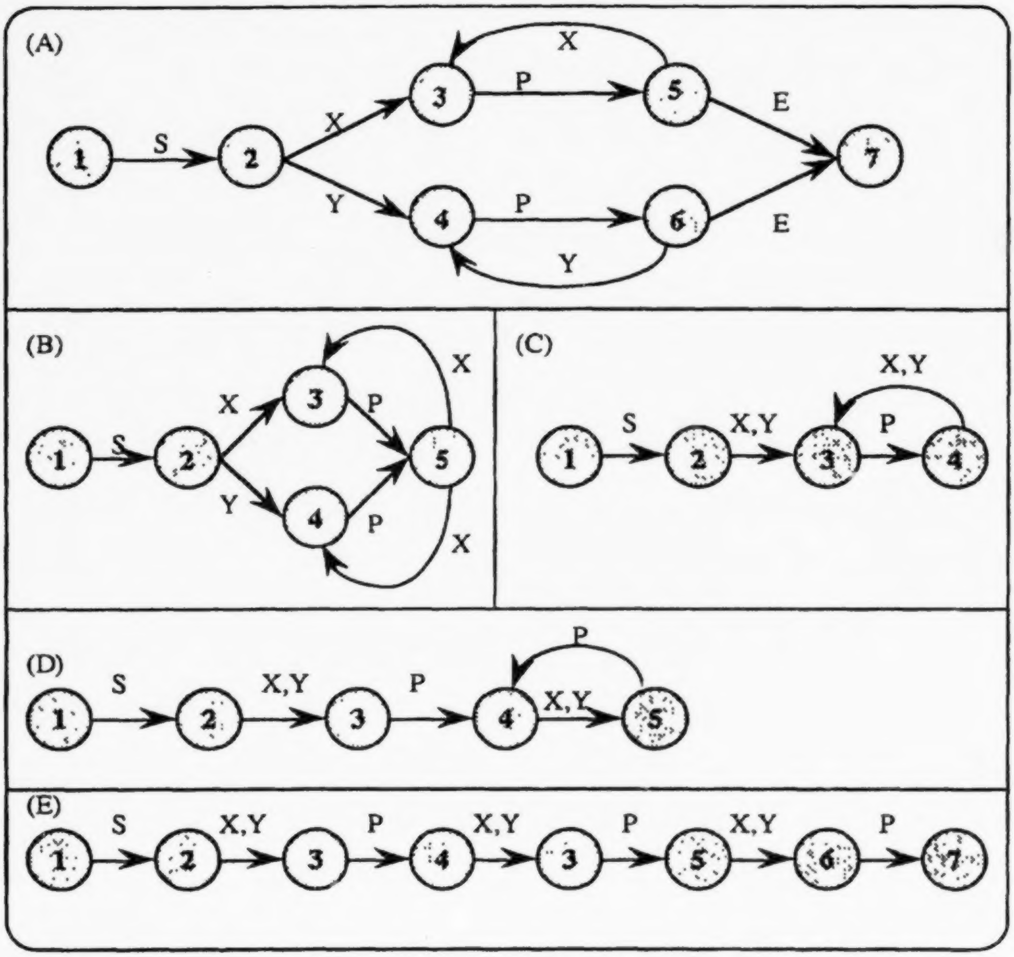

Fig. 5.24 The architecture and results of the first experiment between the SRN and FSRN. (a) This describes the various states that both networks are to learn. Notice that the two paths are different except for the common point seven, which is a consequence of the common prediction, *P*, leading to that state. (b) The initial SRN mapping before training. (c) The mapping of the SRN network after 300 trials, (d) after 700 trials and (e) after 3600 trials (after [MaNo92]).

The sequences used were the following: SPPXE, SPPXPPXE, SPPXPPXPPXE
SPPPXE, SPPPXPPPXE, SPPPXPPPXPPPXE.

In the examples, if *P* appears after *X* then the number of *Ps* appearing after *X* is the same as the number of *Ps* appearing before the first occurrence of *X*. Again, only the

FSRN could predict the correct number of *Ps* following an *X*. The internal representation of the SRN was found to have the same mapping for two *Ps* following an *X* as for three *Ps*. After 300 trials the sequence of *SPPX* and *SPPPX* was encoded the same. But, after 700 trials they were separated correctly and remained separate until 3200 trials. Following this, the two sequences again converged to the same representation and since the network was correctly predicting the next symbol, it never changed. Thus the number of *Ps*, and therefore, the sequence identity was lost. This loss shows the need for the FSRN and its improved context representation approach. As a note, according to the description of *the problem*, in context of this particular experiment, once the two sequences converge they should *not* be able to separate. The only explanation that I can provide for this is that the convergence of the two sequences resulted in a shallow enough minima, which evidently was escapable. In general, Day and Davenport claim that these minima are quite difficult to escape.

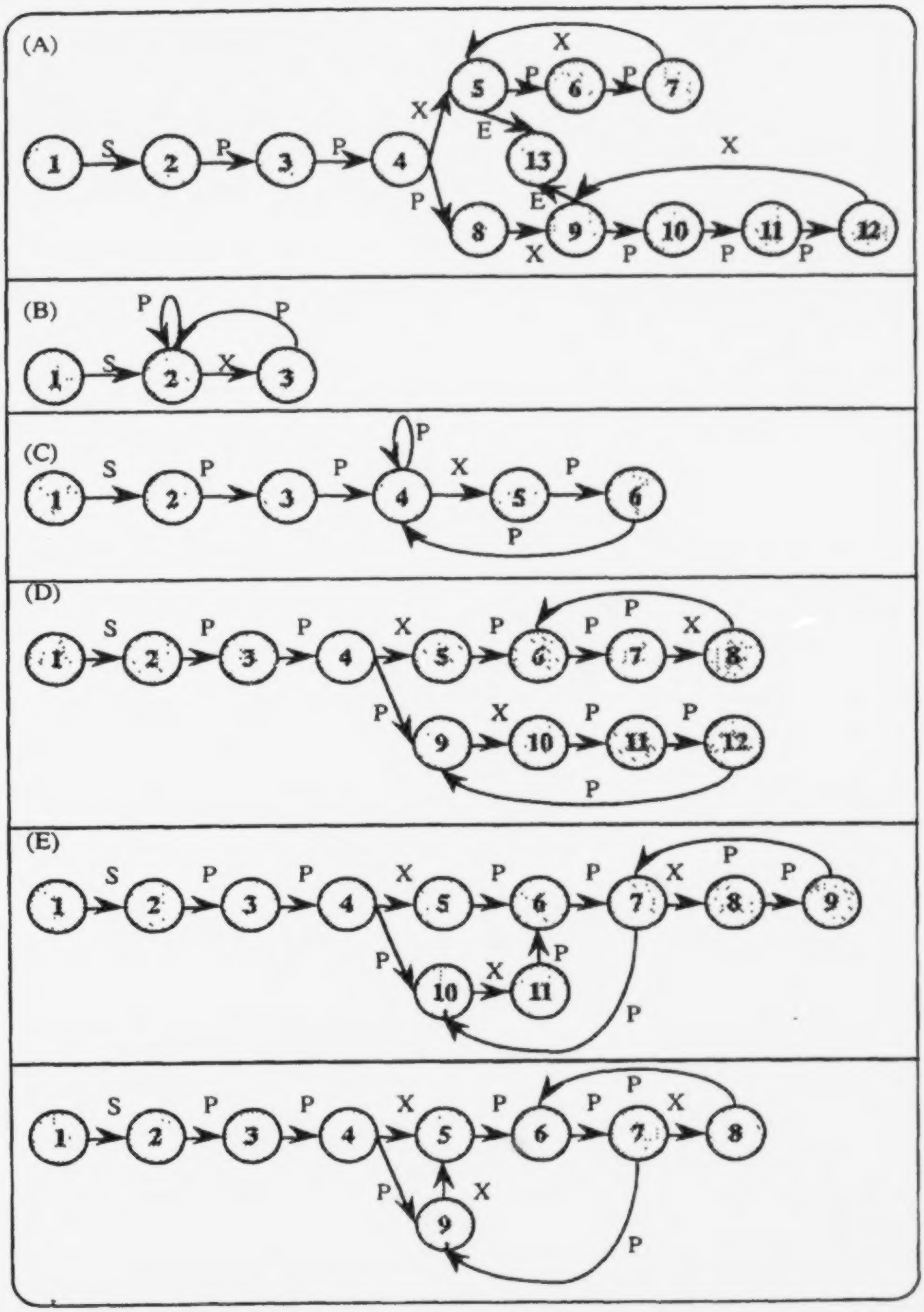

Fig. 5.25 The results of the second SRN experiment. (a) Before training, (b) after 300 trials, (c) after 700 trials, (d) after 1500 trials and (e) after 3200 trials (after [MaNo92]).

5.4 Temporal and Hierarchical Neural Networks

These hierarchical networks use temporal integration to perform temporal pattern recognition, which is necessary for CSR. They do recognition by first applying the patterns to a series of low level, localized modules that recognize fundamental characteristics (phonemes). This layer of modules then passes their results to a series of higher level modules. The level of recognition abstraction is proportional to the number of layers. Therefore, if the lowest level recognizes phonemes, syllables, or demi-syllables then the upper layers could recognize more complex structures such as words, phrases, or even sentences. This ability rests mainly in the design of the isolated module that efficiently manages temporal sequences. In this section, temporal architectures and techniques are discussed and some implementations are provided with the recognition experiment results.

5.4.1 The Hierarchical Network

The temporal module is the entire basis for this network. The temporal module is a complex temporal processing unit. Its internal components are comprised of sequence neurons, and two special neurons - an internal result neuron (IRN) and a linear sum neuron (LSN). These components are connected in a very complex fashion using inhibitory and excitatory connections and using both linear and nonlinear activation functions.

Each sequence neuron is connected to its upstream neighbor (where time propagates from left to right) by an excitatory connection and, it receives inhibitory inputs from all other sequence neurons other than its immediate upstream neighbor. The excitation state of each sequence neuron changes with a fixed time constant, which gives it a fading memory. This interconnection strategy with the fading memory results in the sequence neurons ability to recognize temporal sequences. Now the first sequence neuron (extreme left) can receive an external excitatory connection and the last sequence neuron (extreme right) can send outside the module an excitatory signal. The sequence neurons use a linear activation function for output. After training, a correct input sequence will cause sequential firing along the sequence neurons with the down stream neighbor providing excitatory output from its activation to its upstream neighbor via its lateral connection resulting in a key step toward temporal recognition. For an incorrect temporal sequence, the inhibitory connections will force the sequence neurons to remain near their reset thresholds. The module architecture is shown in Fig. 5.26, although the inhibitory connections have not been shown for clarity.

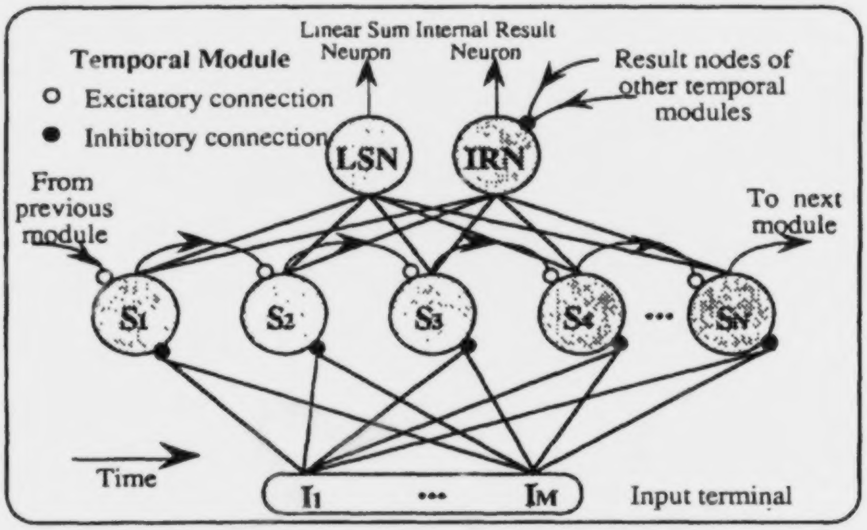

Fig. 5.26 An enhanced temporal module (after [YZTK90]).

The output of each sequence neuron is connected to the input of the module's IRN and LSN by a fixed weight connection. The IRN recognizes a correct event sequence by temporally integrating the sequence nodes' outputs and by using an appropriate bias and time constant. The IRN uses a sigmoid activation function to produce its output. Shortly after the temporal sequence has passed, the IRN will return to its quiescent state. This is how recognition is done by a single module. As one might foresee, this idea can be extended to longer temporal sequences by linking several modules together via the two external excitatory connections mentioned earlier. This approach is discussed in the next sub-section.

Concatenation of Temporal Modules

For longer temporal sequences the recognition does not occur within a module. The module's purpose here is to provide evidence that the portion of the sequence that it observed was correct. This portion constitutes a single event, for instance, a phoneme in a word. In a sense, the module provides a similar function to what the sequence neurons do, since now, many modules are connected in series using the external lateral sequence neuron connections. The IRN neuron of a module is no longer used since it is not responsible anymore for making the recognition decision. The outputs of the LSN neurons from each module, which evidently use a linear activation function, are connected to an external result neuron that uses a sigmoid activation function. The weighted connections between the LSN and ERN are learned using the standard LMS algorithm. The recognition decision of the longer temporal sequence is the sole responsibility of the external result neuron; the lower layer modules only provide evidence for portions of the temporal sequence. Since the problem of recognition has been extended to exist between modules a new problem arises. This involves how the modules will manage an event that occurs between consecutive modules. Can neither module recognize this partial event? If both modules could recognize it, were it centered on their inputs, will the partial stimulation be sufficient to activate it now? If not, will the loss of this event break the temporal sequence resulting in the failure to recognize the complete event? These questions are addressed in the next sub-section.

Frame Overlap

If a temporal event overlaps into the next module, the temporal integration may not suffice to fire enough sequence nodes to activate the IRN or the last sequence neuron, which provides supporting evidence to the next module that a correct temporal sequence exists. Yet, "a module may be activated by a partial sequence, provided the sequence is sufficiently long to exceed the node's activation level" [YZTK90]. If this is not sufficient, a different strategy is used. The module of Fig. 5.26 uses the LSN to generate its output from the weighted sum of the sequence neurons, i.e., it generates a linear sum of the number of correct events seen by the sequence neurons. The temporal integration is performed by the *External Result Neuron* (ERN). This architecture is shown in Fig. 5.27.

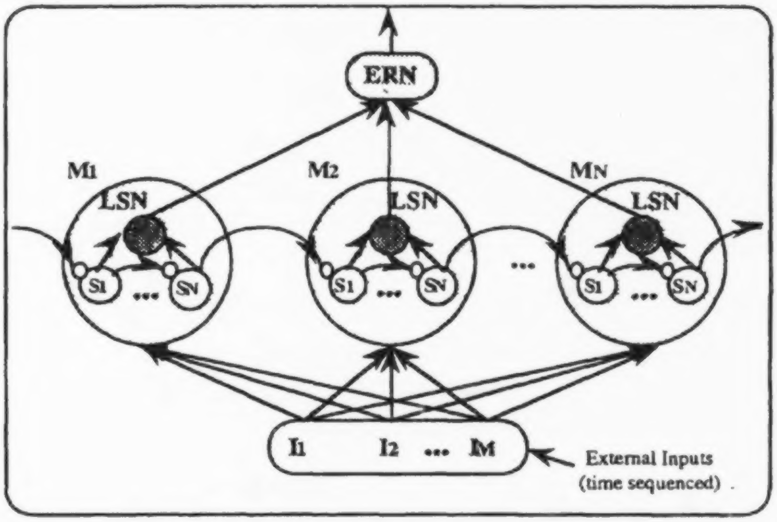

Fig. 5.27 A chain of temporal modules designed to recognize an extended temporal sequence (after [YZTK90]).

Essentially the temporal events are not acknowledged at the module level since they overlap one another and are not isolated to a module. Instead, they are acknowledged at the ERN level where the full temporal event *is* seen. This approach solves the problems posed earlier. The ERN node, which has a sigmoidal output response, replaces the function of the IRN of the individual module (which has a sigmoidal response). The ERN integrator time constant should be based on the minimum length of the module's partial sequence. The idea of these hierarchical units can be taken even further. For instance, Fig. 5.28 shows one possible form where each level-1 module may be represented by the configurations provided either by Fig. 5.26 or Fig. 5.27.

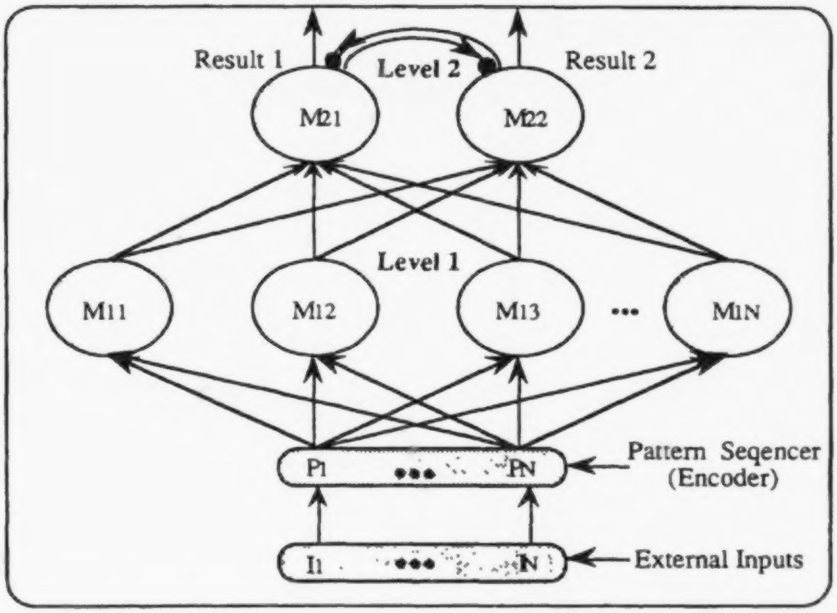

Fig. 5.28 A hierarchical network for extended temporal recognition (after [YZTK90]).

As early abstractions are realized in level-1 (primary processing), level-2 realizes more complex abstractions. For example, if the level-1 modules have architectures like those of Fig. 5.27, phonemes and then words can be recognized. In turn, the words are recognized as sentences by the level-2 modules. In the hierarchy, the modules decrease in number while the abstractions become more complex. Inputs are fed into a decoder that directs the patterns to the correct level-1 module. Modules in the highest level act in a *winner-take-all* fashion. This is done by the intermodule inhibitory connections. Thus, the module that recognizes the temporal sequence inhibits its neighbors within the layer.

Series Connection Experiment

Although [YZTK90] tested their system using a motion detector, many aspects can be applied to the problem of continuous speech recognition. In the first experiment, three modules were cascaded in series, where each module consisted of ten sequence neurons. Modules M_1 through M_3 were trained to recognize patterns I_1 through I_3, respectively. Each pattern consisted of ten events that represented the motion of an object. The ERN was trained to recognise the temporal sequence $\{I_1, I_2, I_3\}$. The authors provide two figures showing the input patterns as seen by the three modules LSN, also the ERN. Input patterns labelled with a p subscript refer to partial input patterns and ones labelled with an x refer to random input patterns. In Fig. 5.29, the sequence $\{I_x, I_{1p}, I_2, I_{3p}, I_x\}$ is given. No significant response is acknowledged for the random patterns. Only after the strong second pattern is shown does the ERN emit significant activity.

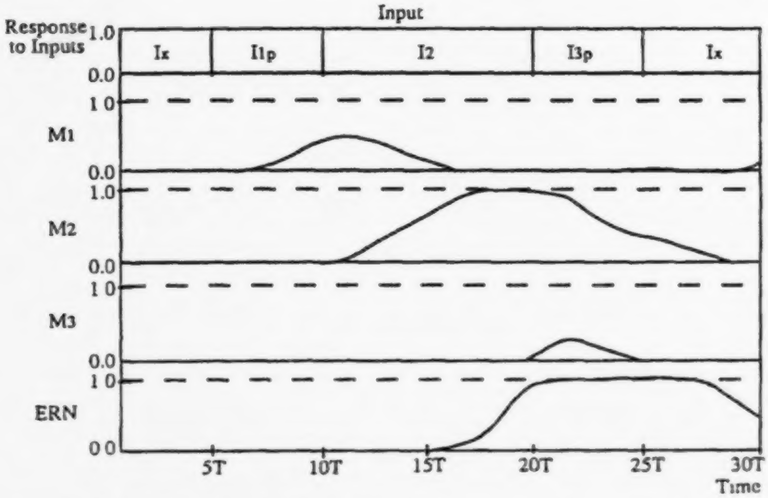

Fig. 5.29 Output of LSNs of M1 through M3 resulting in a correct ERN response. This shows correct temporal recognition as seen by the ERN activation (after [YZTK90]).

The final random pattern is inhibited as seen by the low LSN output activity of the three modules. The ERN finally becomes active upon the presentation of I_{3p}. In the next experiment the previous input sequence is permuted slightly with respect to I_{1p}. The important observation is that the I_{1p} decays to zero when I_2 appears. There is insufficient temporal activity due to the lag between I_{1p} and I_2, the result being that the ERN does not fire. The activation plots for this experiment are shown in Fig. 5.30. In summary, disjoint activations are not usually sufficient for a sequence to become active.

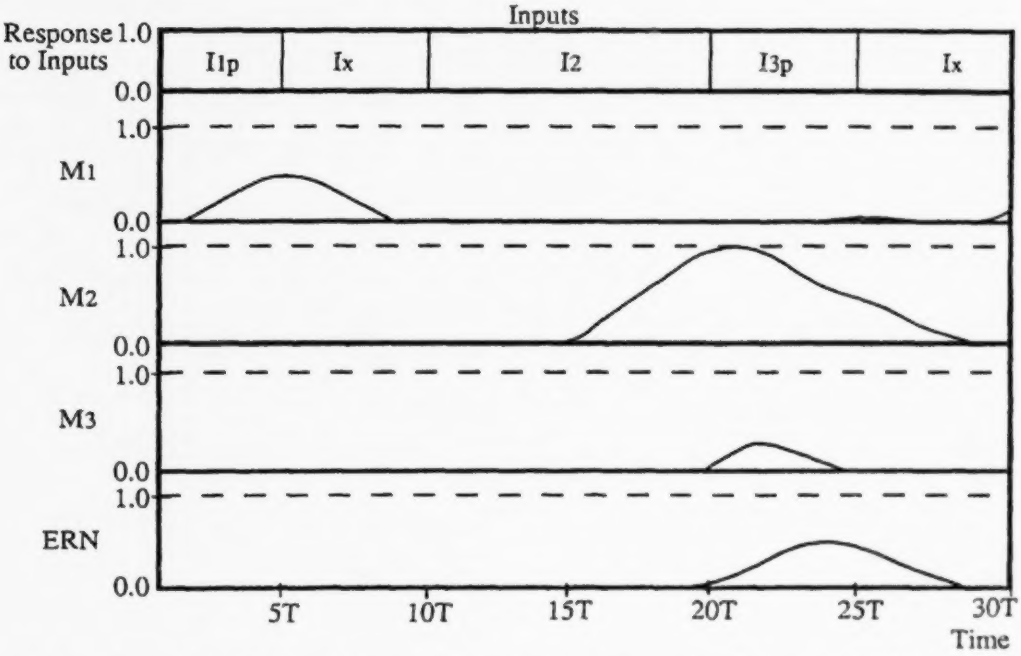

Fig. 5.30 Output of LSNs of M1 through M3 resulting in an incorrect ERN response. Unsuccessful temporal recognition is due to the nonoverlapping patterns of I_{1p} and I_2 as shown by the low activation level of the ERN (after [YZTK90]).

Hierarchical Network Experiment

In this set of experiments, the modules are set up according to Fig. 5.28. "Modules M_{11}, M_{12}, M_{13}, and M_{14} were trained independently to recognize input patterns I_1, I_2, I_3, and I_4, respectively" [YZTK90]. Each temporal sequence consisted of six events. Level-2 modules were then trained to recognize patterns $\{I_1, I_2, I_3\}$ for module M_{21} and $\{I_1, I_2, I_4\}$ for module M_{22}. The first sequence presented to the network was a correct temporal sequence consisting of patterns $\{I_1, I_2, I_3\}$. The resulting activation plots of Fig. 5.31 show the various level-1 (M_{1x}) and level-2 (M_{2x}) outputs. Figure 5.31 clearly shows that as a temporal integration proceeds across the correct events, the correct level-2 module, M_{21}, recognizes the event and becomes active.

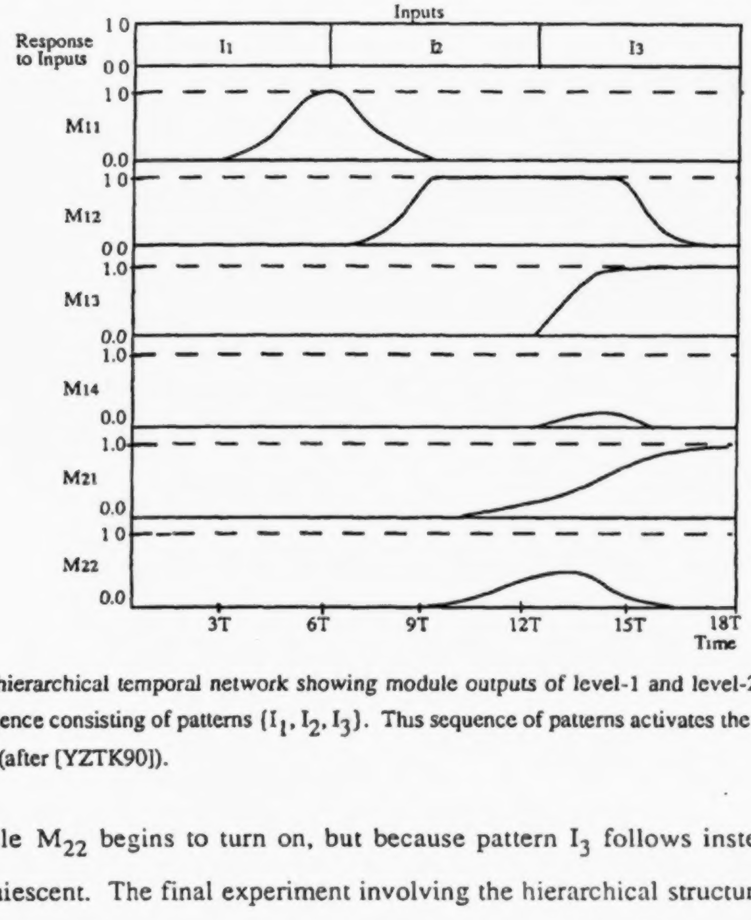

Fig. 5.31 A hierarchical temporal network showing module outputs of level-1 and level-2 for a correct temporal sequence consisting of patterns $\{I_1, I_2, I_3\}$. This sequence of patterns activates the correct output module, M_{21} (after [YZTK90]).

Module M_{22} begins to turn on, but because pattern I_3 follows instead of I_4, it becomes quiescent. The final experiment involving the hierarchical structure is used to show that the level-2 modules can complete a sequence even if part of the cues are missing. Here, the first cue, I_1, is missed but the remainder of the sequence is shown correctly, albeit pattern I_2 is shown only partially. Again, the temporal net successfully activates the correct level-2 module, M_{21}. The module activation plots are shown in Fig. 5.32.

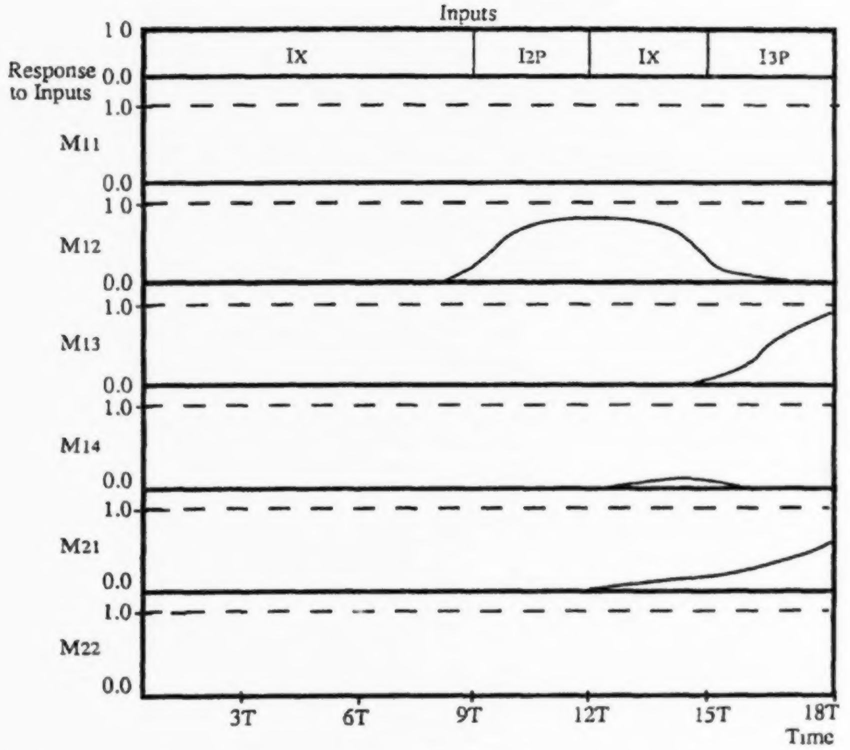

Fig. 5.32 A hierarchical temporal network showing outputs of level-1 and level-2 modules for a partial temporal sequence consisting of patterns $\{I_{2p}, I_3\}$ as shown by output modules M_{21} and M_{22}. This sequence of patterns is to activate module M_{21} (after [YZTK90]).

By the end of the sequence M_{21} has become active though the first cue provided by M_{11} was completely missed. One final property of this temporal neural network is that it is invariant to short bursts of noise. This is evident by reexamining the result exhibited by Figs. 5.29 and 5.32. If a noise burst is applied to the system it is rejected. Correct events that persist (for a long enough time) create a temporal effect permitting the temporal integration to exceed its threshold. A noise burst would likely die away prior to activation of the ERN. "Due to the temporal integration process, a burst of noise appearing at the input of, the network will not undo the results of many successful samples during the recognition process" [YZTK89].

5.4.2 Adaptive Dynamic and Associative Memory Model

The adaptive dynamic and associative memory model's (ADAM) purpose is to recognize speech using a hierarchically structured neural network. Its advantages include background noise rejection, pattern completion, real-time operation through an intricate parallel network, and dynamic learning of new patterns by sprouting memory pathways as needed to form the necessary representations. It uses a special model to steer the data through the correct pathways to improve recognition. The steering process, called *guided propagation,* is a key and unique feature of this model. The actual controller of this process is a special module within the network known as the *control unit* An input to this network, which may be a phoneme or word, is represented by a time based series of discrete spectral speech frames and spatial inputs of the speech. The principal signals types of temporal pattern recognition, to a large extent, involve current and past inputs, context, and prediction.

Guided Propagation

This principle is based on the simultaneous interaction of two processes that act upon the same network nodes. The first process provides stimulus to the context dependent neurons in the time dimension and the other processes affect these neurons spatially. Together, pathways are preactivated showing likely trajectories through the space. This interaction provides the temporal processing found in many successful recognition systems. A process of pattern activation is shown in four stages in Fig. 5.30. In stage one the active neuron (black) slightly preactivates three other neurons (grey). Simultaneously, the feature detectors (square cells at top) are spatially activating other context dependent neurons that favor an observed trajectory. Together, these two independent operations provide a single connected memory pathway through the space that results in the activation of an output unit. The output unit (black square cell at bottom) identifies the utterance. The output units are used as classifiers, i.e., each output unit is tuned to one pattern. Now, this propagation process is controlled by a complex internal network module, the *control unit,* that senses different aspects of the data at different levels of abstraction and helps the propagation of selective pathways based on its observations.

The control units purpose is to "selectively facilitate the propagation of internal signals along the memory pathways, in relationship with the required depth of inferences and to take decisions concerning the sprouting of new internal representations, alternately favoring either learning or robust recognition" [Béro91]. The control unit additionally checks retrieval efficiency using a feedback loop that carries the output results. Lastly, it provides on-line modulation of parameters including rhythm and intensity (prosodic system incorporation). So, the generation of a sentence is seen as a guided process through the network that is modulated by an overseeing control unit that helps the guidance.

The selective activation of an internal memory pathway is dependent on three criteria:
(i) λ – This is used to represent "the bottom-up action of a *feedforward flow* of stimuli . . ." [Béro91]. As seen in Fig. 5.33, the left-to-right direction of propagation is mapped onto the time dimension by feeding the left most nodes with the earliest stimulus of the pattern series and feeding the last component into the right most pathway node.

(ii) β – This represents "the top-down action of a backward facilitation, issued from deeper events" [Béro91]. This flow of information flows against time, right-to-left, in a manner to favor correct activation.

(iii) CD – The activation of these stimuli is locally controlled by *context-dependent* (CD) units. Their purpose is to detect coincidence between stimuli (S), and context (C) that represents the *expected* left-to-right flow. The coincidence of both signals is regulated by the CD unit's internal threshold. The CD unit activates when the temporal and synchronous activity is sufficient and by the favorable backward threshold reduction effect of β.

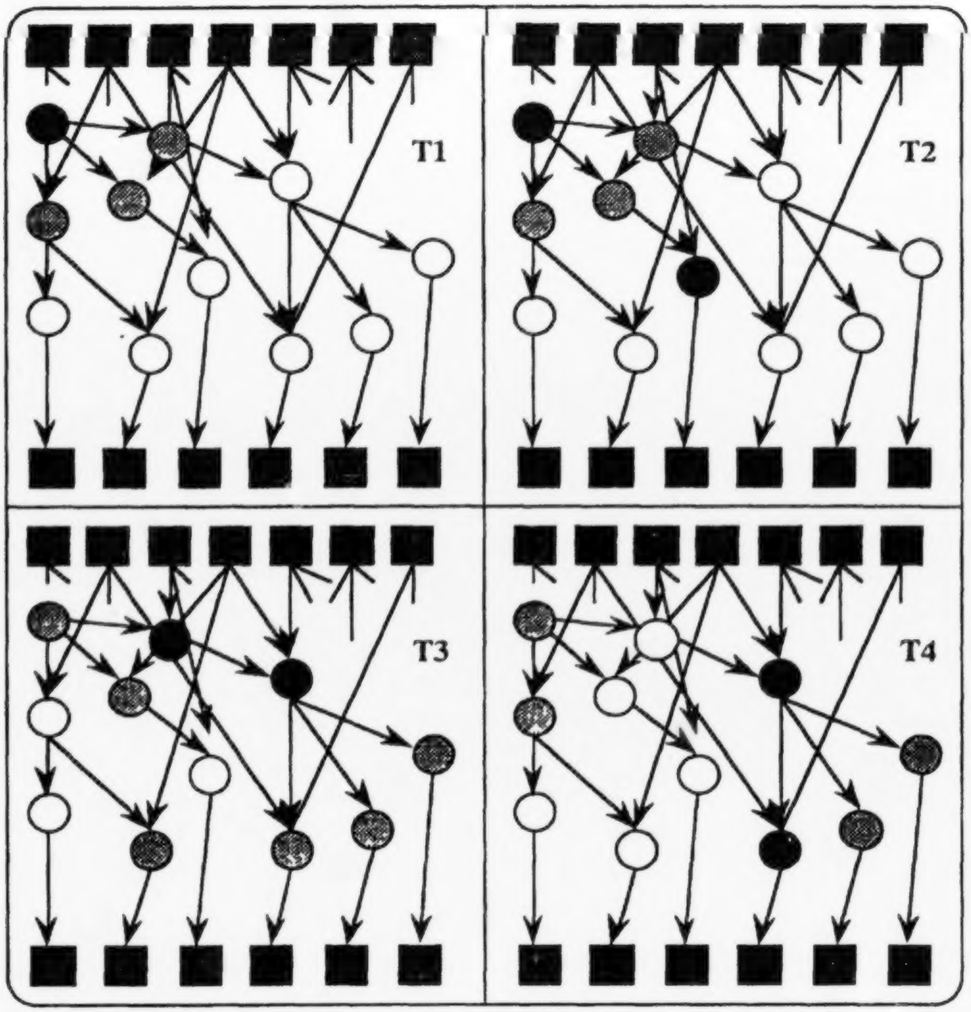

Fig. 5.33 A set a four snapshots in time (T1 to T4) of an utterance's activation. The input units (square neurons at top of network) represent spatial input to the neurons where a blackened square represents active input. The context neurons (circles) receive two input types; spatial from input neurons and time dependent from other neurons, where time progresses from left to right (blackened neurons represent full activation and grey neurons represent partial activation). An output neuron (square nodes at bottom) is activated after the series has been presented and represents the utterance's identity (after [Béro91]).

This description encompasses the internal construction and operation of a *module*. Connections of these modules in a hierarchical fashion can recognize increasingly complex acoustic events. A system of this nature is currently under development, the proposed architecture of which is shown in Fig. 5.34. The shaded regions in the figure have already been implemented in software. In the proposed parallel and serial system, modules are mapped out in layers where each layer represents a knowledge level. Repeating this process at higher levels help to develop an architecture that has syntactic modules tuned to the time dimension and semantic ones tuned to the spatial dimension. These word detectors feed into cluster detectors and semantic feature detectors. The syntactic modules monitor the cluster detectors through a narrow temporal window. These modules, which feed the semantic detectors, have low temporal accuracy because the order of words within a phrase is less important. A layer receives internal event representations of the speech as memory pathway *activations* propagate through the connections to particular units within the layer, i.e., prior modules output nodes stimulate consecutive modules input nodes through learned connections (memory pathways). Reasoning processes such as ***property inheritance*** is used by the semantic network under the watchful eye of the control unit. Property inheritance is a process by where a particular new word is identified as like other known words in a particular aspect. The memory pathways tuned to this aspect are then connected to this new word since this will help establish its identity. For instance, a new word is identified with words that have property ***A***, therefore, it inherits currently established memory pathways with other known words that have property ***A***.

In the perception part of the architecture weak feed forward signals are propagated through the syntactic and semantic modules. This propagation stimulates the pathways connected to the next node, thus developing context and temporal relationships as propagation continues. But these weak signals are used in the activation of words whereas the strong connections established in the production modules are used for the detection of spontaneous data flow. A relationship exists between the generation and perception since spontaneity exists within words.

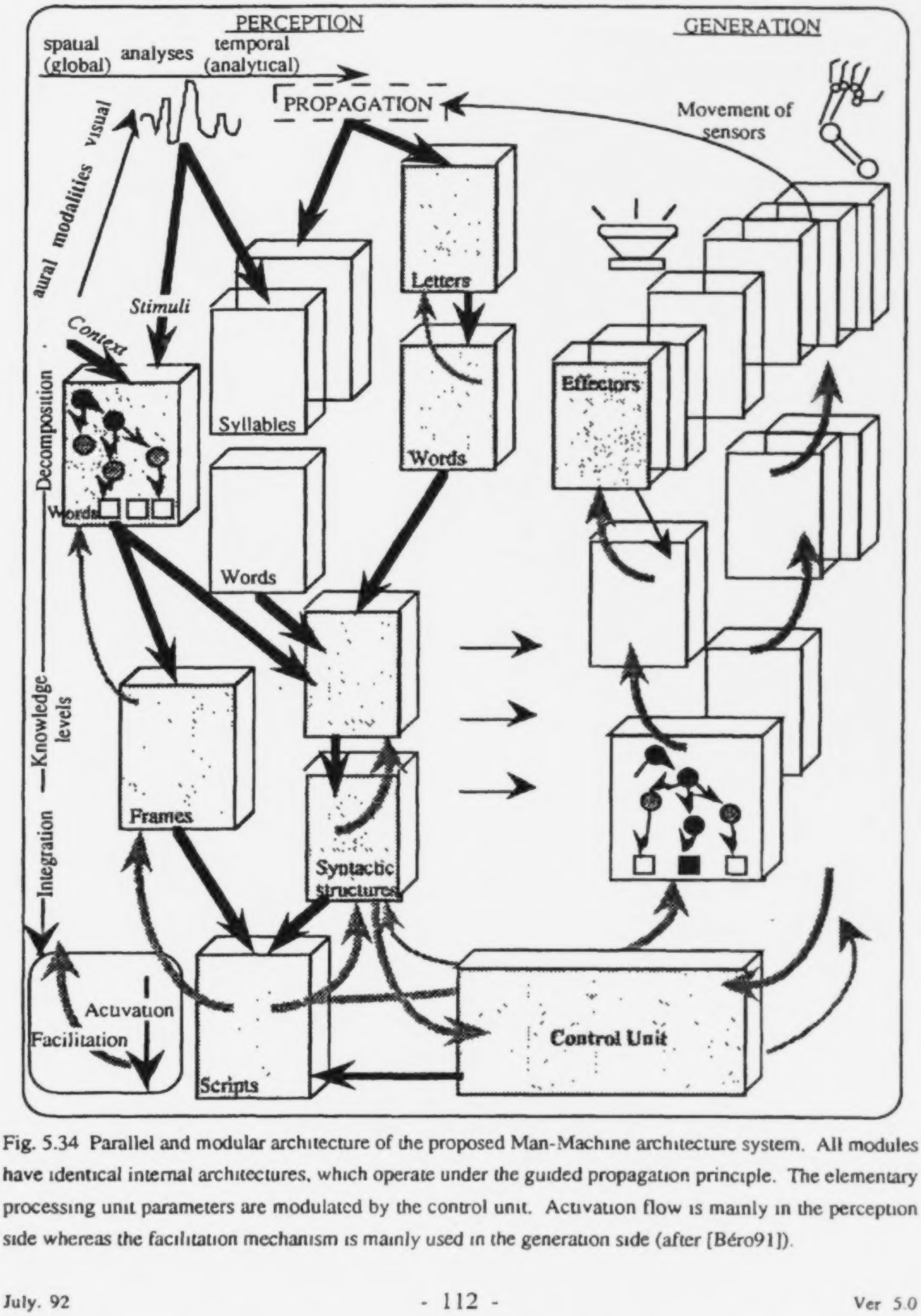

Fig. 5.34 Parallel and modular architecture of the proposed Man-Machine architecture system. All modules have identical internal architectures, which operate under the guided propagation principle. The elementary processing unit parameters are modulated by the control unit. Activation flow is mainly in the perception side whereas the facilitation mechanism is mainly used in the generation side (after [Béro91]).

Thus, a proposed unsupervised learning rule is being developed to help in a cross support approach. "Among a set of random productions, only those that are validated by the perception device are retained" [Béro91]. A full paper describing these proposals is currently under development.

In summary, three different neural network architectures have been investigated to deal with the problem of temporal variations as it pertains to continuous speech recognition. The first architecture involved catenation of modules for the sole purpose of recognizing long temporal sequences. The second architecture involved a hierarchical structure. It was used to recognize increasingly complex abstractions from a continuous stream of speech. Either may be used to recognize phonemes, words, phrases, and possibly sentences. These networks are invariant to bursts of noise and noise in the sequences. They can complete patterns when portions of a pattern from a sequence, or even entire patterns of the sequence, are missed. The third architecture, ADAM, used an intricate set of layers that used both temporal and spatial input to recognize continuous speech. A special process called guided propagation was developed to improve recognition. The propagation process is controlled by a central control unit. Unsupervised learning is also proposed to correlate perception and generation of the speech. This system is proposed to communicate in a Man-Machine interface where dynamic learning is used. The system is currently under development.

5.5 Summary

This chapter described many different approaches to temporal speech recognition using neural networks. The first section looked at TDNN type architectures. Often slight architectural modifications were made that provided added functionality. For instance some networks such as the TFM had hand-tuned architectures that were optimized for the problem. Others claimed modularity and connectivity strategies to be helpful. Still others sensed parameters in the learning process and dynamically adapted their architecture to suit the data set. With temporal BP, a few different models were examined that focused the backpropagated error signal to optimize learning and reduce the network size. This method also performed the backpropagation step during the forward pass by maintaining certain context components. Another method used adaptive time delays and weights to do temporal recognition and used on-line learning to adapt dynamically to new patterns It also claimed that momentum provided improved attractor basins, though results were worse during training. The final section of this chapter examined hierarchical neural networks. These networks are more biologically oriented since modularity plays a large part in their functionality. By this, each module acts almost like a single neuron - only knowing its local responsibility - not knowing the *big picture*. Each module provided evidence for a single acoustic event, like a phoneme for instance. These networks also have time constants built into their neurons, which tune themselves to the sequence lengths. They filter out incorrect sequences by suppressing output activity. If the temporal sequence does not arrive sufficiently close together or last the appropriate length, then the time constants will inhibit the neuron's activation. These networks were designed to recognize patterns at different levels. They are invariant to noisy sequences and are time-shift invariant due to the special module design and weight restrictions. The last hierarchical network that was examined proposed a spatial and temporal approach to recognition. Both input types, with a central control unit that used the guided propagation principle, were able to do temporal pattern recognition.

VI
SURVEY DISCUSSION

6.1 Introduction

The survey discussion is intended both as a review of the material and as a platform from which comparative discussions about the networks are made. Thus, I hope to bring to the attention ideas that are interesting and worthy of further investigation. As described, many approaches contained in Chapter V have great merit and expectations for doing high accuracy continuous speech recognition. Not all the approaches have even been thoroughly tested. The reason for discussing these *beta* systems was done primarily to bring to the attention the innovative thinking that is on-going in this area and to stimulate possibly further attempts to improve the technology.

6.2 Discussion

To review briefly, the survey objectives were to find what was occurring in the area of temporal speech recognition using neural networks. Specifically, the objective was to examine the NN techniques that work well with temporally segmented and corrected speech and those techniques that work well with natural speech, i.e., techniques that recognize a continuous flow of natural speech. Some approaches sample periodic segments of the incoming waveform, perform spectral analysis, and finally present this to the net. The net is responsible for dealing with everything after that point. This includes coarticulation effects, segmentation, time-shift invariance, timing differences, etc.. Now in the other camp, the network is provided, say, temporally decomposed segments. It is not responsible for doing the temporal decomposition, only the recognition. Based on these criteria, the probable benefits and difficulties of each side are examined. Prior to this focus discussion, reasons and motivation leading up to the development of temporal neural networks is given.

In the past, researchers expressed a need to recognize natural speech. After realizing our limits in computing power and technical knowledge about processing and analyzing speech we settled into the now, well developed isolated speech recognition domain. Here, many insurmountable problems were greatly reduced that included intonation, stress, coarticulation, and segmentation. Often, even greater restrictions were imposed like noise free speech, both for training and testing. Further, the speech was restricted to come from only one speaker and that this speaker remained in the same emotional state, both during training and testing! After about twenty to thirty years, excellent recognition results were achieved exceeding 98%, even for large vocabularies. During this time though technology vastly improved in the area of computing power and knowledge of speech feature analysis. With these improvements came the resurgence of neural networks that demanded the use of both areas' new advancements. Logically, scientists and engineers alike, applied their knowledge of the isolated speech recognition domain to the neural networks and found limited success. But the isolated NN recognizers and the feature analysis techniques imposed an upper limit on what could be done. They turned back to the speech data to figure out how more discriminatory information could be squeezed out of the data and how to reduce the blockades built by coarticulation and segmentation. The realization of these *end-of-the-road* limitations caused a new NN era to emerge, the results of which may be significantly closer to our objective of natural continuous speech recognition.

Instead of working against the problems of coarticulation, pitch, stress, etc., researchers examined these issues from a new perspective, which was to ask themselves if these properties can help the system? Other engineers and scientists developed ways to temporally decompose speech. This achievement greatly reduced the problem of coarticulation and segmentation, which always posed a built-in error margin to recognition systems. The engineers realized that new architectures were necessary to maintain context and memory if temporal continuous speech recognition was to succeed. Modularity and connectivity issues developed in which sub-networks recognized local low-level acoustic events. In turn, these results were passed to higher modules in the hierarchy that recognized more complex structures. This approach permitted recognition of acoustic events such as words and phrases. Other networks stressed time-shift invariance and frequency-time-shift invariance. *On-the-fly* learning techniques allowed new patterns to be dynamically incorporated to existing network structures. Network algorithms were improved to the point where they could detect whether to add a new connection, and its location in the NN. The networks also knew where to prune poorly used connections. Network parameters no longer needed to be tweaked through human intervention since these were dynamically and optimally adjusted by the network, often during the learning process. Hybrid architectures were introduced that included both new and old paradigms.

In effect, what is happening today is a tendency to develop closer and closer models of what we perceive to be occurring in the biological domain. Granted some implementations used to solve these problems are not truly biologically plausible, but the spark of ingenuity surely stemmed from this knowledge. Artificial neural network speech recognizers are frequently using more high-level acoustic, syntactic, and grammatical speech knowledge to help make more complex and informed recognition decisions.

Presegmenting the speech alleviates some problems for these recognizers in one sense but creates others. If the speech is temporally segmented then coarticulation between words is reduced, yet, this does not say anything about context, stress or intonation and how these should be modified. On the other hand, if the recognizer learns the above features continuously, it must contend with segmentation. To find word boundaries the network must maintain a history, since often, the next word starts before the previous one ends, and vice versa, i.e., overlapping boundaries can occur at either end of a word.

The Neural Network Discussion

Time-delay neural networks advantage include its ability to provide time-shift invariance to the speech through special weight structure restrictions. Also, its learning procedure does not require precise temporal pattern alignment. This approach has physical time delays built into its architecture. The delays are not truly dynamic since they do not change their values according to the data or network parameters. The delay weights are learned but the magnitudes of the delays are not learned. Here, a pattern presented two time steps ago (a delay of two) would become active during the third pattern sequence. This is a problem since the architecture, including the delays, may not be optimum. If the time delays are too long, excessive context will be maintained and the network will learn relationships between patterns that may not be desired (artifacts). It will not become tuned to the correct set of sequences. Also, *where* the time delay units appear in the architecture is important. Sometimes, the number, location, and magnitude of the time delays, also the general architecture size, is motivated by *intuition*. This is the weak link and leads to nonoptimum architectures, which in turn, leads to poor solutions. Two problems exist: (i) if the network is too large, the number of degrees of freedom encourages artifacts and noise to be retained in the weight space and prevents the network from extracting the regularities that result in development of the desired generalization characteristic, and (ii) if the network is too small, even the regularities cannot be held due to insufficient degrees of freedom. In both cases, reduced recognition results.

Taking the above TDNN and modifying its architecture and algorithm result in encouraging improvements leading to significant advantages. If, for instance, the network also maintains frequency-time-shift invariance then the network has a distinct advantage over the standard TDNN because frequency-time-shift invariance absorbs speech variations in different speakers and their different speaking manners, i.e., it makes different speakers sound more alike. Time-shift invariance makes pattern locations on the input appear less altered. This would be advantageous in both cases when we consider temporally segmented speech since we need not worry so much about how we present the data as what its properties are.

If there is a block from the lower layers focusing its activation to a unit in the upper layers of the TDNN, another advantage is realized. First, a block is two-dimensional; one dimension being time (rows), the other, frequency (columns). The additional advantage is that the NN captures global features in the upper layer due to the connectivity structure that performs temporal integration. This type of modified TDNN can recognize words (98.2%), short phrases (84.1%), long phrases (83.9%), and continuous speech (82.8%) of the time, which always exceeds that of a standard TDNN. The typical input to these networks is FFT frames but TD frames can easily be substituted since they are similar to LPC coefficients, which are also of a spectral nature and often used with TDNNs.

A final instance of the TDNN investigated dynamic architectures. Obviously, this approach appears to guarantee optimal architectures that greatly reduce the two problems mentioned above. This is done through adaptive delays, adaptive weights, and adaptive Gaussian shaped input windows. Processes that add, delete, and split neurons comprise the network optimization techniques. The combination of Gaussian shaped input windows and the standard TDNN weighting structure provided the time-shift invariance of these networks as well as providing a better input context representation at the utterance boundaries (Gaussian shape).

The next section concerns temporal backpropagation. As described in Chapter II, BP in its standard format cannot do temporal recognition because it lacks the means of maintaining activation history. But by using context units, which amount to special recurrent weighted links, a finite amount of activation history is maintained, turning a one-shot network into a temporal one. Temporal BP networks can use temporally segmented speech. This is the case since BP is a feedforward supervised method that requires a teacher present during training. For instance, given that there are four temporally decomposed acoustic sequences comprising an utterance, then some teacher (output pattern) must be present during this sequence of presentations to help in forming the correct trajectories in the weight space resulting in the desired vector at the output layer.

Focused BP is a good approach to temporal speech recognition because it uses context recurrency with some decay to perform temporal recognition, and by using *activity traces* (which are derived from the context units decay factor and error signal) the weights can be updated in the *forward* pass. This is clearly advantageous since it reduces the computational load on the network and thus reduces the training time. It is important to note that to prevent consecutive acoustic event sequences from being correlated during learning, acoustic events (which are composed of an ordered sequence of temporally decomposed events) *must* be randomly permuted during network presentation. This is because the *same* context weights are used for all training data. An advantage of recurrent nets is that their outputs depend on the input signal for almost any length of time. Still, the recurrent network training procedures typically suffers from high computational and storage demands.

Another good approach to temporal BP adapts both the weights and time delays using gradient descent and uses either epoch or on-line updates. The adaptive time delays have an advantage over fixed delays found in standard TDNN architectures because adaptive delays allow the network to discover simpler and more accurate mappings. This type of network operates where both input and output are presented continuously. This makes it unnecessary to segment training data. Training, here, is done by presenting equal length *trajectories* (sets of ordered, temporally decomposed events). The delays replace the need for any input buffer or recurrency relationship (as mentioned above) to provide the history component. On-line learning has an advantage over epoch because the weight changes do not accumulate to large values that evidently cause the gradient to stray due to quantization of the updates. It was also found that the use of momentum in these schemes provided better recognition results, not necessarily better training results. This was because momentum had the effect of reducing the high frequency oscillations of the instantaneous gradient (since on-line learning is being used here) about its mean value (true gradient). It also filters out quantization noise due to discrete parameter updates if epochal learning is used.

Any temporal BP network that uses context within the network, whether through delays or context units, has many advantages over a straight BP network that is unfolded in time using a replicated input layer acting as a set of delays. The main advantage is that temporal type connections, which are in amongst the internal network, learn regularities in the same way as the hidden units.

The last section involved hierarchical neural networks. These networks appeared particularly interesting due to their unique architectural arrangements. The hierarchy is intended to simulate the different levels of perception, i.e., low-level abstractions are fed into higher modules that recognize increasingly complex acoustical arrangements. Their distinct advantage is just that - the recognition is not *one-shot* - it is incremental and as such, can realize a large variety of acoustical structures. It is like trying to learn all English words or, learning only the phonemes that *can* form all the words; the latter is of course easier but the additional step is to learn how to recognize correctly the phonemes to form the words. This type of network functions well using temporally segmented frames of speech (phonemes, words, phrases), i.e., a temporally decomposed utterance that is separated (using TD, FFT, or LPC) into a discrete set of related patterns. These networks can fill gaps from phrases that have been missed, are invariant to certain kinds of noise, and are able to reject pattern sequences that are very far off the norm. These characteristics are very powerful and useful in natural speech recognition. If there is too much noise or if the acoustic sequence is altered *we* may not even recognize it. Also, if the temporal stimuli are not present with the correct timing, again, we may not recognize the sequence so we can hardly expect a simplifed NN to do better than ourselves. This approach has not yet been extensively tested as it was introduced only late in 1990 [YZTK90]. It appears though, very biologically oriented. It uses time constants in the modules to perform temporal integration; a necessary component for any CSR. It also uses lateral inhibition to inhibit incorrect sequences from continuing and lateral excitation to lower the thresholds of neurons in the propagation pathway of correct sequences. This model is also time-shift invariant and maintains temporal context using time constants within each module. Its strong point is that it is very good at recognizing long sequences of events such as words, phrases, etc.. This type of neural network has successfully been implemented in silicon.

Another recognition approach is to take smaller networks trained for sub-tasks, i.e., one net specializes in recognizing vowels, while another specializes in recognizing consonants, and connect them to a common higher level *blank* network. This network, with additional *glue* units, allows the successful union of these two independently trained networks. The glue units help to inhibit the wrong network from becoming active while exciting the correct sub-network. This approach is similar in function to Michael Jordan's gating nets. This is necessary since the pattern is presented to *all* sub-networks simultaneously since we do not know at this point to which sub-network the temporal segment belongs. This scheme is unfortunately a *glue and patch* approach and results in nonoptimal architectures, although the authors feel it is faster to train for equal size tasks. These networks use temporally segmented utterances that have to be time-aligned to be time-shift invariant. With the nonoptimal architectures come the problems mentioned in prior discussions. Various time constant schemes need to be established in the upper layer(s) for recognition of a complex sequence (word, phrase) since partial stimuli must be maintained until the identity can be correctly determined. First, there is no *real* temporal recognition happening here. Sub-modules are not communicating to each other except through the glue units that are added until *it works*. The authors' comments suggest the approach is very unstable since training parameters must be carefully chosen for satisfactory recognition, thus some intuition is mandatory for the problem.

The last hierarchical network model used a complex form of feedback to learn the temporal sequences. This network was driven by both temporal and spatial inputs. The neurons themselves exhibited contextual properties, which helped to recognize the temporal sequences. In addition to these characteristics, the entire system was modular. This modularity meant that the sequences features were learned by the network such that the more complex structures found in the upper layers came as a result of activation of less complex structures of lower layers. Thus, a layer, excluding the final output layer, only provided a localized portion of information. It did not, for instance, recognize a word, but more likely did recognize a smaller unit like a phoneme. This entire operation was overseen by a central controller which helped to establish favorable memory pathways to the sequences while extinguishing less favorable ones. This process of selective propagation as employed by the central control unit was justly called, guided propagation. This system (ADAM), as a whole, appears to have a significant amount of biological motivation underlying its operation. It is still under development.

VII
CONCLUSIONS AND RECOMMENDATIONS

The work described in this survey was motivated by the need to extend the understanding of temporal continuous speech recognition systems which use artificial neural networks. A study was conducted that began by first evaluating earlier technologies and approaches, which resulted in the identification of their limitations in the continuous speech recognition domain. Motivation was then provided to shift toward more biologically plausible techniques. This resulted in the presentation of temporal decomposition, which was determined as a good alternative to segmenting speech and provided an excellent feature. Some approaches were proposed to enhance the frame information content and to help in determining better boundaries of the acoustic events. The focus of the survey discussed many temporally based NN speech recognition systems that operated on either temporally decomposed samples or continuous spectral frames.

The results of the research into this area appear to be encouraging. In particular, the evaluation of many systems were done, which included the three broad classes of TDNN, TDBP, and hierarchical temporal networks. These classes proved that limited continuous temporal speech recognition is possible with recognition in the high 80% range for many systems, even above 90% for particular systems and speech tasks. Each system focused on particular problems of the continuous recognition domain and often emerged with success.

A main objective was to evaluate the pros and cons of using a NN to perform both segmentation and recognition simultaneously (and implicitly) as opposed to a network that recognized only temporally segmented speech (and possibly enhanced and/or corrected speech).

Based on the study it appears that networks that recognize temporally decomposed speech frames and have either enhanced and/or corrected those data are generally superior to networks that *implicitly* do both segmentation and recognition. This conclusion is based upon the following observations:

(i) The temporal decomposition technique, originally introduced by Atal, has been perfected to the point where each acoustic event has been tailored accurately with respect to length, location, and representation. Redundancy and misrepresentation have been minimized. And, being temporal decomposition, coarticulation has been well modelled. The result being that a continuous stream of naturally spoken speech can be accurately decomposed into its constituent acoustic components.

It has not been shown to what degree a *NN* can decompose a natural stream of speech or how well the network represents the information - or for that matter, being represented in the hidden units, how the decomposition is performed or what the decomposition is based on.

(ii) Regular network architectures appear traditionally to be well suited to recognize static spatialized patterns. Still, in our case we now ask that the network recognize a sequence of patterns. This request has been answered by using context, recurrency, and delays, used within the various architectures. The result is that these networks can recognize a sequence of patterns since they maintain a history. The particular implementation technique of this purpose varies widely but the objective is identical - recognize a temporal sequence. Those networks can be tailored entirely to the representation of temporal events than also to worrying about how to achieve implicit segmentation.

(iii) Also, NNs appear to be well suited to the speech representation problem since they easily manage linear and nonlinear logic, which speech exhibits. As a note however, some tasks are not easily separable using a unified MLP due to the discrimination boundaries. A hierarchical network can more easily construct such boundaries because of its physical separation into subnetworks. This advantage is significant and should be used when designing systems where the speech tasks have previously found difficulties with MLPs.

(iv) Given that events have been temporally segmented now allows us to perform secondary processing on each segment. This could include time normalization, pitch translations or contour corrections, and amplitude normalization. In a system that does all steps simultaneously, the system must first address the segmentation problem that has already been discussed. Following this, still, there is the problem of whether the network will trigger on the first acoustic segment of the event. If it is missed, the entire segment may be missed because context only begins to build upon the successful identification of the first component of a known sequence. Fortunately, a few hierarchical networks only require a partial stimulus of the initial segment for context to be triggered, as follows.

(v) Another distinct advantage of the separated system is that if the first component of the temporally decomposed sequence is missed, the remaining components may *still* provide sufficient context to cause recognition. This is the case for the hierarchical networks that recognized sequences where components were missing, corrupted, and interspersed with burst noise.

As a recommendation, it would appear that further research into the combination of temporal decomposition and hierarchical neural networks would be beneficial. The temporal decomposition process seems to provide excellent segmentation, superior coarticulation modelling, and good discriminatory information based on natural speech. Hierarchical networks seem to have the most to offer temporal sequence recognition. This combination has not yet been fully explored but appears to be an optimum mix that should yield very rewarding results. Also, some enhanced TDNN architectures appear to be promising and may yield very good results when used with TD, in particular the FTDNN and BWTDNN models.

REFERENCES

[Atal83] B. Atal, "Efficient coding of LPC parameters by temporal decomposition," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 81-84, 1983.

[BaMA89] G. Bailly, P. Marteau, and C. Abry, "A new algorithm for temporal decomposition of speech: Application to a numerical model of coarticulation," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 508-511, 1989.

[BCDM88] F. Bimbot, G. Chollet, P. Deleglise, and C. Montacie, "Temporal decomposition and acoustic-phonetic decoding of speech," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 445-447, 1988.

[Béro91] D. Béroule, "The adaptive, dynamic and associative memory model: An actual present tool for vocal human-computer communication," personal communication to MMT@DCIEM.dnd.ca, 1991. (See also M. Taylor, F. Neel, and D. Bouwhis, *The Structure of Mulitmodal Dialogue*, Elseiver Science: North Holland, 1989, pp. 189-202.

[BoWa91] U. Bodenhausen and A. Waibel, "Learning the architecture of neural networks for speech recognition," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 117-120, 1991.

[Bris86] G. Bristow, *Electronic Speech Recognition*. London, England: Collins, 1986, 395 pp.

[BrAl91] K. Brown and R. Algazi, "Speech recognition using dynamic features of acoustic subword spectra," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 293-296, 1991.

[Chur87] K. Church, *Phonological Parsing in Speech Recognition* Norwell, MA.: Kluwer Academic Publishers, 1987, 261 pp.

[CDKK87] Y. Chow, M. Dunham, O. Kimball, M. Krasner, G. Kubula, J. Makhoul, P. Price, S. Roucos, and R. Schwartz, "BYBLOS: The BBN continuous speech recognition system," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 89-92, 1987.

[DaDa91] S. Day and M. Davenport, "Continuous time temporal backpropagation with adaptable time delays," (unpublished - obtained off of *News comp.ai.neural*) *submitted to IEEE trans. on Neural Networks*, August 1991.

[DARP88] Defense Advanced Research Projects Agency, *DARPA: Neural Network Study*. Fairfax, VA.: AFCEA International Press, 1988, 629 pp.

[DBMC88] P.Deleglise, F. Bimbot, C. Montacie, and G. Chollet, "Temporal decomposition and acoustic-phonetic decoding for recognition of continuous speech," *Proc. IEEE 9th International Conference on Pattern Recognition*, pp. 839-841, 1988.

[FeKi91] K. Ferens and W. Kinsner, "Word synthesis using fuzzy splicing," *Proc. IEEE Western Canada Conference on Computer, Power, and Communications Systems in a Rural Environment (WESCANEX)*, pp. 210-215, 1991.

[FuMa91] Y. Fukuda and H. Matsumoto, "Phoneme recognition using recurrent neural networks," *2nd European Conference on Speech Communication and Technology (Eurospeech '91)*, pp. 1419-1422, 1991.

[GoHa91] Y. Gong and P. Haton, "Nonlinear vector interpolation by a neural network for phoneme identification in continuous speech," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 121-124, 1991.

[HaFW91] P. Haffner, M. Franzini, and A. Waibel, "Integrating time alignment and neural networks for high performance continuous speech recognition," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 105-108, 1991.

[Hube89] D. Huber, "A statistical approach to the segmentation and broad classification of continuous speech into phrase-sized information units," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 600-603, 1989.

[Huck90] M. Huckvale, "Exploiting speech knowledge in neural nets for recognition," *IEEE Speech Communication*, vol. 9, no. 1, Feb. 1990.

[ItFu90] T. Ito and K. Fukushima, "Recognition of spatio-temporal patterns with a hierarchical neural network," *International Joint Conference on Neural Networks (IJCNN)*, pp. 273-276, 1990.

[Jord86] M. Jordan, "Serial order: A parallel distributed processing approach," Tech. Rep. ICS Rep. 8604, Inst. for Cognitive Science, University of California, San Diego, La Jolla, CA., 1986.

[Juan91] B. Juan, "Speech recognition in adverse environments," *Computer speech and language* vol. 5, pp. 275-294, 1991.

[KaSc85] E. Kandel and K. Schwartz, *Principles of Neural Science*. New York, NY: Elsevier, 979 pp., 1985.

[KlSt72] D. Klatt and K. Stevens, "Sentence recognition from visual examination of spectrograms and machine-aided lexical searching," *IEEE Proceedings of the Speech Communication and Processing*, pp. 315-318, 1972.

[Komo91] Y. Komori, "Time-state neural networks (TSNN) for phoneme identification by considering temporal structure of phonemic features," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 125-128, 1991.

[Koho88] T. Kohonen, "The 'neural' phonetic typewriter," *Computer Magazine*, pp. 11-22, March 1988.

[Kuhn87] G. Kuhn, "A first look at phonetic discrimination using a connectionist network with recurrent links," Tech. Rep. SCIMP Work. Paper no. 4/87, Inst. for Defense Analyses, Communications Research Division, IDA-CRD Log no. 82018, 1987.

[Lea73a] W. Lea, "Evidence that stressed syllables are the most readily decoded portions of speech," *The Journal of the Acoustic Society of America*, (abstract only), Oct. 1973.

[Lea73b] W. Lea, "An algorithm for locating stressed syllables in continuous speech speech," *The Journal of the Acoustic Society of America*, (abstract only), Oct. 1973.

[Lea73c] W. Lea, *Trends in Speech Recognition*. Englewood Cliffs, NJ: Prentice-Hall, 1980.

[LHHM89] K. Lee, H. Hon, M. Hwang, S. Mahajan, and R. Reddy, "The SPHINX recognition system," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 445-448, 1989.

[Lipp87] R. Lippmann, "An introduction to computing with neural nets," *IEEE Acoustics, Speech, and Signal Processing Magazine*, pp. 4-22, April 1987.

[LjRi91] A. Ljolje and M. Riley, "Automatic segmentation and labelling of speech," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 473-476, 1991.

[Lube88] D. Lubensky, "Learning spectral-temporal dependencies using connectionist networks," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 418-421, 1988.

[MaNo92] A. Maskara and A. Noetzel, "Forcing simple recurrent neural networks to encode context," *Proceeding of the 1992 Long Island Conference on Artificial Intelligence and Computer Graphics*, (not yet passed), 1992.

[MBML89] G. Mercier, D. Bigorgne, L. Miclet, L. Le Guennec, and M. Querre, "Recognition of speaker-dependent continuous speech with KEAL," *IEEE Proceedings I: Communications, Speech, and Vision*. vol. 136, no. 2, pp. 145-154, 1989.

[MHBK91] N. Morgan, H. Hermansky, H. Bourlard, P. Kohn, and C. Wooters, "Continuous speech recognition using PLP analysis with multi-layer perceptrons," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 49-52, 1991.

[NaHa88] S. Nakagawa and Y. Hashimoto, "A method for continuous speech segmentation using HMM," *Proc. IEEE 9th International Conference on Pattern Recognition*, pp. 960-962, 1988.

[NaCC89] M. Nasri, G. Caelen-haumont, and J. Caelen, "Using prosodic rules in speech recognition expert system," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 671-674, 1989.

[MaFa89] M. Niranjan and F. Fallside, "Temporal decomposition: A framework for enhanced speech recognition," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 655-658, 1989.

[McRu89] J. McClelland and D. Rumelhart, *Explorations in Parallel Distributed Processing*. Cambridge, MA: MIT Press, 1989, 344 pp.

[Pars86] T. Parsons, *Voice and Speech Processing*. New York, NY: McGraw-Hill, 1986, 402 pp.

[RaKL91] P. Ramesh, S. Katagiri, and C. Lee, "A new connected word recognition algorithm based on HMM/LVQ segmentation and LVQ classification," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 113-116, 1991.

[RaSc78] L. Rabiner and R. Schafer, *Digital Processing of Speech Signals*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1978, 512 pp.

[ReMM91] S. Renals, D. McKelvie, and F. McInnes, "A comparative study of continuous speech recognition using neural networks and hidden Markov models," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 369-372, 1991.

[Sawa91] H. Sawai, "Frequency-time-shift-invariant time delay neural networks for robust continuous speech recognition," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 45-48, 1991.

[Tomi86] M. Tomita, "An efficient word lattice parsing algorithm for continuous speech recognition," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 1569-1572, 1986.

[TWPS91] J. Tebelskis, A. Waibel, B. Petek, and O. Schmidbauer, "Continuous speech recognition using linked predictive neural networks," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 61-64, 1991.

[Waib88] A. Waibel, *Prosody and Speech Recognition*. San Mateo, CA: Morgan Kauffmann Publishers, 1988, 212 pp.

[Waib89a] A. Waibel, "Modular construction of time-delay neural networks for speech recognition," *Neural Computation*, vol. 1, pp. 39-46, 1989.

[Waib89b] A. Waibel, "Phoneme recognition using time-delay neural networks," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 3, pp. 328- 339, 1989.

[Waib91] A. Waibel, "Evaluation of speaker-independent phoneme recognition on TIMIT database using TDNNs," *2nd European Conference on Speech Communication and Technology (Eurospeech '91)*, pp. 105-108, 1991.

[Watr90] R. Watrous, "Phoneme discrimination using connectionist networks," *Journal of Acoustical Society of America*, vol. 87, no. 4, pp. 1753-1772, 1990.

[Watr91] R. Watrous, "Source decomposition of acoustic variability in a modular connectionist network," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 129-131, 1991.

[WWHL91] J. Wang, C. Wu, C. Haung, and J. Lee, "Integrating neural nets and one-stage dynamic programming for speaker independent continuous mandarin digit recognition," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 69-72, 1991.

[YZTK89] T. Yeap, S. Zaky, J. Tsotsos, and H. Kwan, "A neural network for temporal pattern recognition," *Proc. IEEE International Symposium on Circuits and Systems*, pp. 778-781, 1989.

[YaHu87] E. Yannakoudakis and P. Hutton, *Speech Synthesis and Recognition Systems*. Toronto, Ont: John Wiley & Sons Canada Limited, 1987, 184 pp.

[Vand89] A. Van Dijk-Kappers, "Comparison of parameter sets for temporal decomposition," *Speech Communication*, vol. 8, pp. 203-220, 1989.

[VaMa89] A. Van Dijk-Kappers and S. Marcus, "Temporal decomposition of speech," *Speech Communication*, vol. 8, pp. 125-135, 1989.

[YZTK90] T. Yeap, S. Zaky, J. Tsotsos, and H. Kwan, "A hierarchical neural network for temporal pattern recognition," *Proc. IEEE International Symposium on Circuits and Systems*, pp. 2967-2970, 1990.

DOCUMENT CONTROL DATA SHEET

1a. PERFORMING AGENCY DCIEM	2 SECURITY CLASSIFICATION UNCLASSIFIED Unlimited distribution -
1b. PUBLISHING AGENCY DCIEM	

3. TITLE

(U) A survey of temporal techniques applied toward neural network based continuous speech recognition

4. AUTHORS

Chris D. Love

5. DATE OF PUBLICATION July 1 , 1992	6. NO. OF PAGES 132

7. DESCRIPTIVE NOTES

8. SPONSORING/MONITORING/CONTRACTING/TASKING AGENCY

Sponsoring Agency:
Monitoring Agency:
Contracting Agency :
Tasking Agency:

9. ORIGINATORS DOCUMENT NO. Technical Report 1992-078	10. CONTRACT GRANT AND/OR PROJECT NO	11. OTHER DOCUMENT NOS.

12. DOCUMENT RELEASABILITY

Unlimited distribution

13. DOCUMENT ANNOUNCEMENT

Unlimited announcement

14. ABSTRACT

(U) Neural network (NN) architectures for the recognition of continuous speech are reviewed in this report. Historically, NNs were developed for the recognition of static patterns. To use such networks for speech recognition required that the speech be segmented into chunks such as words or phonemes that could be recognized individually as static patterns. In real speech, the execution of a work of a phoneme depends to some extent on what words or phonemes precede it. These coarticulation effects cause problems unless prior history is used ot aid the recognition process. New architectures are being developed to permit the speech stream to be treated as the continuous stream that it is. Segmenting still occurs, which is legitimate, since humans do identify individual words, syllables and phonemes, but the segmentation may be intrinsic to the recognition process. Alternatively, the segmentation may be done by a front-end process that preserves coarticulation effects. Hierarchic structures that recognize events of increasing temporal scale seem to provide the most promising path toward effective recognition of continuous speech.

15. KEYWORDS, DESCRIPTORS or IDENTIFIERS

(U)

515971

CA011347